REDUCING HUMAN LABOR COST IN DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

A Dissertation
Presented to
The Academic Faculty

By

Haoming Jiang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

May 2021

REDUCING HUMAN LABOR COST IN DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Thesis committee:

Dr. Tuo Zhao, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Yao Xie
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Diyi Yang
School of Interactive Computing
Georgia Institute of Technology

Dr. Chao Zhang
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Weizhu Chen
Microsoft Azure AI
Microsoft

Date approved: January 1, 2020

Multivac: “THERE IS AS YET INSUFFICIENT DATA FOR A MEANINGFUL ANSWER.”

Isaac Asimov, “The Last Question”

For my parents, sister and the love of my life

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Tuo Zhao for continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I will never forget the days and nights we spent together working on the challenging research problems.

I am also grateful to all of my thesis committee members: Weizhu Chen, Yao Xie, Diyi Yang, and Chao Zhang. Their constructive suggestions to my thesis research are very important for me to finish my dissertation.

Besides my thesis committee, I want to thank the H. Milton Stewart School of Industrial and Systems Engineering and Machine Learning program. The students, faculty and staffs have provided me with a wonderful study and work environment. I benefited a lot from unforgettable lectures delivered by Shabbir Ahmed, Turgay Ayer, Santanu Dey, Yajun Mei, Arkadi Nemirovski, Hongyuan Zha, and Huan Xu. My thanks also go to Santanu, Alan Erera, Justin Romberg, Amanda Ford and Stephanie Niebuhr for the help in my first and graduate semester, and go to George Family Foundation for their financial support through Wally George fellowship. I would also like to thank ARC Student Fellowships and Amazon Student Fellowship for supporting my research.

In addition, I would like to thank Weizhu, Wei Wei, and Bing Yin for offering me the internship opportunities in their groups and supporting me working on exciting projects. I will never forget all the wonderful memories when interning in their groups.

I am very fortunate to have worked with my amazing collaborators and co-authors: Tianyu Cao, Zhehui Chen, Minshuo Chen, Bo Dai, Siawpeng Er, Xinyu Fei, Jianfeng Gao, Jason Ge, Jawei Han, Pengcheng He, Mingyi Hong, Lingkai Kong, Xingguo Li, Yan Li, Chen Liang, Wenjing Liao, Xiaodong Liu, Liyuan Liu, Feng Liu, Han Liu, Jie Lyu, Wendi Ren, Qianli Shen, Yuyang Shi, Yifu Sun, Mengdi Wang, Zhaoran Wang, Rui-

jia Wang, Chong Wang, Dingding Wang, Yujia Xie, Mengjiao Yang, Yue Yu, Dangqing Zhang, Yuchen Zhuang and Simiao Zuo. The collaborations with them have prominent impact on my research and benefit me a lot.

I must thank all of my friends, labmates and colleagues: Alexander Bukharin, Xinshi Chen, Binghong Chen, Tianlong Chen, Xiaohan Chen, Wenbo Fei, Hongyang Gao, Yilin Guo, Yandong Li, Zhan Lin, Weiyang Liu, Mingyi Liu, Tianyi Liu, Ding Pei, Tao Shen, Ruolin Su, Kaiyue Wang, Guanyi Wang, Ethan Wang, Yi Wang, Yanzhao Wu, Liyan Xie, Fenghao Xu, Yuanzhe Xu, Tingting Xuan, Jiacheng Yang, Henry Shaowu Yuchi, Wanrong Zhang, Rui Zhang, Yujie Zhao, Miao Zhen, Zhen Zhong, and Shixiang Zhu.

Last but not least, I want to thank my parents for their love and care during my pursuit of a Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

BOND **B**ERT-Assisted **O**pen-Domain **N**ame Entity Recognition with **D**istant Supervision

COSINE **Co**ntrastive **S**elf-Training for F**ine**-Tuning Pre-trained Language Model

ENIGMA **E**valuati**N**g d**I**alo**G** syste**M**s **A**utomatically

NEEDLE **N**oise-aware w**E**akly sup**E**rvise**D** continua**L** pr**E**-training

NER Named Entity Recognition

NLP Natural Language Processing

NLU Natural Language Understanding

SMART **SM**oothnessinducing **A**dversarial **R**egularization and b**R**egman p**R**oximal poin**T** op**T**imization

SUMMARY

Deep learning has fundamentally changed the landscape of natural language processing (NLP). The success of deep learning techniques relies on huge amounts of manually labeled data in many applications. However, large amounts of labeled data are usually prohibitive or expensive to obtain. In addition, accurately evaluating the NLP models also requires expansive human evaluation. This dissertation focuses on reducing such human labor cost in deep learning for NLP. We develop novel frameworks for training deep learning models with limited/noisy annotation and a novel framework for estimating human evaluation scores.

Training with Limited Supervision. Many state-of-the-art models are first pre-trained on a large text corpus and then fine-tuned on downstream tasks. However, due to limited data resources from downstream tasks and the extremely high complexity of pre-trained models, aggressive fine-tuning often causes the fine-tuned model to overfit the training data of downstream tasks and fail to generalize to unseen data. To address such an issue in a principled manner, we propose a new learning framework for robust and efficient fine-tuning for pre-trained models to attain better generalization performance. The proposed framework contains two important ingredients: 1. Smoothness-inducing regularization, which effectively manages the complexity of the model; 2. Bregman proximal point optimization, which is an instance of trust-region methods and can prevent aggressive updating. Our experiments show that the proposed framework achieves new state-of-the-art performance on a number of NLP tasks including GLUE, SNLI, SciTail and ANLI.

Training with Weak Supervision. When manually labeled data is not available, we can leverage domain expert knowledge to generate weakly labeled data. The weak supervision, though does not require large amounts of manual annotations, yields highly incomplete and noisy weak labels via external knowledge bases. To address this challenge, we propose a new computational framework, which leverages the power of pre-trained

language models (e.g., BERT and RoBERTa) to improve the prediction performance of NLP models. Specifically, we propose a two-stage training algorithm: In the first stage, we adapt the pre-trained language model to the downstream tasks using the weak labels, which can significantly improve the recall and precision; In the second stage, we drop the weak labels, and propose a self-training approach to further improve the model performance. Thorough experiments on benchmark datasets demonstrate the superiority of the proposed framework.

Dialogue Evaluation without Human Interaction. In addition to the model training, we also address the problem of reliable human-free automatic evaluation for dialog systems. An ideal environment for evaluating dialog systems, also known as the Turing test, needs to involve human interaction, which is usually not affordable for large scale experiments. To bridge such a gap, we propose a new framework named ENIGMA for estimating human evaluation scores based on recent advances of off-policy evaluation in reinforcement learning. ENIGMA only requires a handful of pre-collected experience data, and therefore does not involve human interaction with the target policy during the evaluation, making automatic evaluations feasible. More importantly, ENIGMA is *model-free* and *agnostic to the behavior policies* for collecting the experience data, which significantly alleviates the technical difficulties of modeling complex dialogue environments and human behaviors. Our experiments show that ENIGMA significantly outperforms existing methods in terms of correlation with human evaluation scores.

CHAPTER 1
INTRODUCTION

Tremendous progresses have been made by deep learning in many natural language processing (NLP) application, such as sentiment analysis [1], named entity recognition (NER, [2]), natural language understanding (NLU, [3]), dialogue generation [4]. However, training an over-parameterized neural network requires large amounts of labeled data. For example, training an English-French neural machine translation model requires 41 millions of parallel sentences (WMT14 dataset [1]). Manually labeling such a dataset is very expansive and sometime prohibitive (e.g., for some applications requiring special domain expertise). Moreover, evaluating an NLP model for some particular applications also needs to involve human interaction. For example, to evaluate the performance of a dialogue agent, a human evaluator needs to talk the agent and give his/her judgment on the agent's fluency, sense of humor, etc. Therefore, how to develop neural network systems without intensive human labor becomes one of the core problems in natural language processing, and it is still actively studied in different applications.

In this thesis, we present novel frameworks for training/evaluating deep learning models with less human labor cost. Specifically, in Chapter 2, we first propose a transfer learning approach with regularized optimization for learning with limited manually labeled data. In Chapter 3 and Chapter 4 , we propose a transfer learning approach with self-training for learning with weakly labeled data, where the weak labels are automatically generated from heuristic rules or domain expert systems. In Chapter 5, we propose an automatic evaluation approach for dialogue systems which is based on off-policy evaluation and can accurately estimate human evaluation scores.

[1] http://www.statmt.org/wmt14/translation-task.html

1.1 Background

In this section, we first review the popular transfer learning approach and the state-of-the-art transformer-based pre-trained models for Chapter 2. We also review the weak supervision for Chapter 3 and Chapter 4, and dialogue systems and dialogue evaluation for Chapter 5.

1.1.1 Transfer Learning and Transformer-based Pre-trained Model

Transfer learning considers the scenario, where we have limited labeled data from the target domain for a certain task, but we have relevant tasks with a large amount of data from different domains (also known as out-of-domain data). The goal is to transfer the knowledge from the high-resource domains to the low-resource target domain. Here we are particularly interested in the popular two-stage transfer learning framework [5]. The first stage is pre-training, where a high-capacity model is trained for the out-of-domain high-resource relevant tasks. The second stage is fine-tuning, where the high-capacity model is adapted to the low-resource task in the target domain.

For many applications in NLP, most popular transfer learning methods choose to pre-train a large language model, e.g., ELMo [6], GPT [7] and BERT [8]. The most popular transfer learning methods are based on transformer models. The transformer models were originally proposed in Vaswani et al. [9] for neural machine translation. Their superior performance motivated Devlin et al. [8] to propose a bidirectional transformer-based language model named BERT. Specifically, Devlin et al. [8] pre-trained a masked language model using a large corpus without any human annotation through unsupervised learning tasks. Such a language model can capture general semantic and syntactic information that can be further used in downstream NLP tasks. The language model is particularly attractive, because it can be trained in a completely unsupervised manner with huge amount of unlabeled data, which are extremely cheap to fetch from internet nowadays. For example, the well-known "Common Crawl Project"[2] is extracting text data from web pages at a rate of about

[2] commoncrawl.org

20TB per month from a variety of domains. The resulting extremely large multi-domain text corpus allows us to train huge language models. To the best of our knowledge, by far the largest language model, GPT-3, has an enormous size of about 175 billion parameters [10]. Many follow-up works further improve the pre-training by introducing new unsupervised learning tasks [11, 12, 13], enlarging model size [14, 15], enlarging training corpora [16, 11, 15] and multi-tasking [17, 18]. All these research for improving pre-training require huge amounts of computing resource.

For the second fine-tuning stage, researchers adapt the pre-trained language model to the target task/domain. They usually replace the top layer of the language model by a task/domain-specific sub-network, and then continue to train the new model with the limited data of the target task/domain. Such a fine-tuning approach accounts for the low-resource issue in the target task/domain, and has achieved state-of-the-art performance in many popular NLP benchmarks [8, 16, 11, 14, 12, 15, 3]. Naive fine-tuning on the limited labeled data with the large pre-trained language model often leads to undesirable generalization performance (e.g., overfitting, and forgetting). To prevent overfitting, existing heuristics include choosing a small learning rate or a triangular learning rate schedule, and a small number of iterations, and other fine-tuning tricks mentioned in [19, 20, 21, 22].

1.1.2 Weak Supervision

Despite the success of transfer-learning with transformer-based pre-trained language model, one bottleneck for fine-tuning LMs is the requirement of labeled data. When labeled data are scarce, the fine-tuned models often suffer from degraded performance, and the large number of parameters can cause severe overfitting [23].

To relieve the label scarcity bottleneck, one approach to fine-tune the pre-trained language models is using only weak supervision. While collecting large amounts of clean labeled data is expensive for many NLP tasks, it is often cheap to obtain weakly labeled data from various weak supervision sources, such as semantic rules [24]. For example, in

sentiment analysis, we can use rules `'terrible'`→`Negative` (a keyword rule) and `'* not recommend *'`→`Negative` (a pattern rule) to generate large amounts of weak labels.

Fine-tuning language models with weak supervision is nontrivial. Excessive label noise, e.g., wrong labels, and limited label coverage are common and inevitable in weak supervision. Although existing fine-tuning approaches [25, 26, 27] improve LMs' generalization ability, they are not designed for noisy data and are still easy to overfit on the noise. Moreover, existing works on tackling label noise are flawed and are not designed for fine-tuning LMs. For example, Ratner et al. [28], Varma and Ré [29], and Mallinar et al. [30] use probabilistic models to aggregate multiple weak supervisions for denoising, but they generate weak-labels in a context-free manner, without using LMs to encode contextual information of the training samples [31]. Other works [32, 33] focus on noise transitions without explicitly conducting instance-level denoising, and they require clean training samples. Although some recent studies [24, 34] design labeling function-guided neural modules to denoise each sample, they require prior knowledge on weak supervision, which is often infeasible in real practice.

Self-training [35, 36] is a proper tool for fine-tuning language models with weak supervision. It augments the training set with unlabeled data by generating pseudo-labels for them, which improves the models' generalization power. This resolves the limited coverage issue in weak supervision. However, one major challenge of self-training is that the algorithm still suffers from error propagation—wrong pseudo-labels can cause model performance to gradually deteriorate.

1.1.3 Dialogue Systems and Dialogue Evaluation

Building dialog systems that can communicate unhindered with humans in natural language has been one of the most important goals of artificial general intelligence research since the 1950's [37]. One of the fundamental research bottlenecks for developing such dialog

systems falls in evaluation, namely how to measure the performance of these systems in an automatic and scalable manner. Different from supervised natural language understanding tasks (e.g., text classification and machine translation), an ideal environment for evaluating dialog systems, also known as the Turing test, involves multi-turn human interaction [37, 38, 39, 40]. While online platforms such as Amazon Mechanical Turk can provide human-based evaluation, they are often expensive and not scalable[41].

Researchers have adopted language quality metrics for single-turn response generation given a fixed context (e.g., BLEU score and perplexity) to automatically evaluate dialog systems [42, 43, 44, 45, 41]. However, these metrics only weakly correlate to human evaluation in practice [38, 39]. One cause of such weak correlation is that language quality metrics rely on the exact match between generated text and ground-truth, which generally do not fully overlap. While certain embedding-based metrics have been developed to combat this lack of coverage [46, 47], they are only post-hoc judgments based on static experience data, and does not necessarily reflect the dynamic quality of multi-turn interactive dialog well [39]. Moreover, evaluation of goal-oriented dialog systems should be based on how well dialog systems collect information from users and whether the goal is completed; language quality metrics are thus unable to meet these requirements.

To overcome the limitations of the aforementioned static evaluation methods, another line of work has proposed to model the interactive process of a conversation as a Markov decision process (MDP) [48, 49, 50, 51, 39, 52]. Accordingly, automatic evaluation of dialog systems can be formulated as an off-policy evaluation (OPE) problem, where a human subject is the so-called "environment" in the reinforcement learning (RL) literature. For instance, Wei et al. [4] propose a model-based approach for goal-orient dialog systems. They first learn an environment/human model from the experience data consisting of human response, and then evaluate a dialog agent/policy by executing the policy within the learned environment. This procedure is known as "self-play evaluation". Such a model-based approach requires accurate estimation of an environment/human when both input

and output are in a *combinatorially* large space, i.e., the trained model needs to be able to mimic complex human behavior of generating meaningful sentences from huge vocabulary. Unfortunately, such a requirement is far beyond the current capability of model-based reinforcement learning algorithms. As a result, evaluations that rely on accurate modeling of the environment is often unreliable. A similar model-based approach is proposed [39] to evaluate open-domain chit-chat dialog systems. In addition to modeling human behavior, they also model the reward function (for mimicking the complex mechanism behind human ratings) based on handcrafted features, which makes evaluation even more unreliable.

1.2 Outline of the Thesis and Main Contributions

This thesis focuses on reducing human labor cost in deep learning for NLP from the following three aspects:

1. Chapter 2: Training with limited supervision.
2. Chapter 3 and Chapter 4: Training with weak supervision.
3. Chapter 5: Automatic evaluation for dialogue systems without human interaction.

1.2.1 Chapter 2: Training with limited supervision

In Chapter 2, we consider improve the fine-tuning stage of the transfer learning with limited manually labeled data. To address the issues of overfitting, aggressive update, and knowledge forgetting mentioned in Section 1.1.1, we propose a new learning framework for robust and efficient fine-tuning on the pre-trained language models through regularized optimization techniques. Specifically, our framework consists of two important ingredients for preventing overfitting:

1. To effectively control the **extremely high complexity** of the model, we propose a *Smoothness-inducing Adversarial Regularization* technique. Our proposed regularization is motivated by local shift sensitivity in existing literature on robust statistics.

Such regularization encourages the output of the model not to change much, when injecting a small perturbation to the input. Therefore, it enforces the smoothness of the model, and effectively controls its capacity [53].

2. To prevent **aggressive updating**, we propose a class of *Bregman Proximal Point Optimization* methods. Our proposed optimization methods introduce a trust-region-type regularization [54] at each iteration, and then update the model only within a small neighborhood of the previous iterate. Therefore, they can effectively prevent aggressive updating and stabilize the fine-tuning process.

We compare our proposed method with several state-of-the-art competitors proposed in [55, 18, 16, 14, 15] and show that our proposed method significantly improves the training stability and generalization, and achieves comparable or better performance on multiple NLP tasks. We highlight that our single model with 356M parameters (without any ensemble) can achieve three state-of-the-art results on GLUE, even compared with all existing ensemble models and the T5 model [15], which contains 11 billion parameters. Furthermore, we also demonstrate that the proposed framework complements with SOTA fine-tuning methods [18] and outperforms the T5 model.

1.2.2 Chapter 3 and Chapter 4: Training with weak supervision

In Chapter 3, we study the named entity recognition (NER) problem under weak supervision. The weak supervision, though does not require large amounts of manual annotations, yields highly incomplete and noisy weak labels via external knowledge bases as mentioned in Section 1.1.2. To address this challenge, we propose a new computational framework based on transfer learning and self-training, which learns accurate named entity taggers from weak supervision without any restriction on the domain or the content of the corpora. To address the challenges in learning from weak supervision, our approach leverages the power of pre-trained language models (e.g., ELMo [6], BERT [8], XLnet [11]) which have

strong expressive power to capture general semantics and syntactic information effectively. To fully harness the power of pre-trained language models for tackling the two challenges, we propose a two-stage training framework. In the first stage, we fine-tune the RoBERTa model [16] with weakly-matched labels to essentially transfer the semantic knowledge in RoBERTa, which will improve the quality of prediction induced from weak supervision. Then we use the RoBERTa model to predict a set of pseudo soft-labels for all data. In the second stage, we replace the weakly-matched labels with the pseudo soft-labels and design a *teacher-student* framework to further improve the recall. The *student* model is first initialized by the model learned in the first stage and trained using pseudo soft-labels. Then, we update the *teacher* model from the *student* model in the previous iteration to generate a new set of pseudo-labels for the next iteration to continue the training of the *student* model. This *teacher-student* framework enjoys the merit that it progressively improves the model confidence over data. In addition, we select samples based on the prediction confidence of the *student* model to further improve the quality of soft labels. In this way, we can better exploit both the knowledge base information and the language models and improve the model fitting. We conduct comprehensive experiments on 5 datasets for named entity recognition tasks with weak supervision. Our proposed method significantly outperforms state-of-the-art weakly supervised NER competitors in all 5 datasets (4 of which by significant margins).

We also study the semi-weakly-supervised setting, where both small manually/strongly labeled and large weakly labeled data are available. Our experimental results show that the propsoed framework significantly improves the model performance on the E-commerce query NER tasks and Biomedical NER tasks. In particular, we achieve new SOTA F1-scores on 3 Biomedical NER datasets: BC5CDR-chem 93.74, BC5CDR-disease 90.69, NCBI-disease 92.28. We also extend the proposed framework to the multi-lingual setting. Note that, the proposed framework can be easily extended to other NLP tasks.

In Chapter 4, we study more NLP tasks under weak supervision and develop a con-

trastive self-training framework to further improve the performance. Specifically, contrastive self-training regularizes the feature space by pushing samples with the same pseudo-labels close while pulling samples with different pseudo-labels apart. Such regularization enforces representations of samples from different classes to be more distinguishable, such that the classifier can make better decisions. To suppress label noise propagation during contrastive self-training, we propose confidence-based sample reweighting and regularization methods. The reweighting strategy emphasizes samples with high prediction confidence, which are more likely to be correctly classified, in order to reduce the effect of wrong predictions. Confidence regularization encourages smoothness over model predictions, such that no prediction can be over-confident, and therefore reduces the influence of wrong pseudo-labels. Our framework is flexible and can be naturally extended to semi-supervised learning, where a small set of clean labels is available. Moreover, since we do not make assumptions about the nature of the weak labels, our framework can handle various types of label noise, including biased labels and randomly corrupted labels. Biased labels are usually generated by semantic rules, whereas corrupted labels are often produced by crowd-sourcing. Extensive experiments on 6 NLP classification tasks using 7 public benchmarks verifying the efficacy of our method. We highlight that our model achieves competitive performance in comparison with fully-supervised models on some datasets, e.g., on the Yelp dataset, we obtain a 97.2% (fully-supervised) v.s. 96.0% (ours) accuracy comparison.

1.2.3 Chapter 5: Automatic evaluation for dialogue systems without human interaction.

In Chapter 5, we study automatic evaluation for dialogue systems without human interaction. we propose a general off-policy evaluation (OPE) framework named ENIGMA for estimating human evaluation score (i.e., how a human would rate a dialog system). Different from the model-based approaches mentioned in Section 1.1.3, which rely on complex modeling of human behavior given combinatorially large vocabulary, ENIGMA takes ad-

vantage of recent advances in model-free OPE and avoids direct modeling of dynamic transitions and reward functions in a complex environment. Moreover, ENIGMA overcomes several limitations of existing OPE methods in order to evaluate dialog systems:

(I) Existing OPE methods only apply to infinite or fixed horizon settings (where horizon length corresponds to number of turns in a conversation), while onversations, on the other hand, often have varying horizon lengths;

(II) Existing OPE methods require experience data to sufficiently cover states and actions a target policy might visit. Due to limited experience data and the combinatorial nature of languages, such a requirement can hardly be satisfied in dialog evaluation;

(III) Certain OPE methods rely on accurate estimation of the behavior policies used to collect the experience data. Unfortunately, such behavior policies are humans or complex dialog systems, and estimating their probabilistic model is essentially a challenging imitation learning problem.

To address **(I)**, we propose a pseudo state padding method, which augments each conversation into infinitely many turns and yet preserves the original policy value; to address **(II)**, we leverage pre-trained language models [8], which essentially transfer knowledge from out-of-domain data to alleviate the coverage issue; to address **(III)**, we adopt a stationary distribution correction estimation approach [56], which directly models the state-action density ratio between the experience data and the target policy [57], and is therefore agnostic to the behavior policy.

We conduct thorough experiments on evaluating goal-oriented (AirDialog, [4]) and chit-chat (ConvAI2, [58]) dialog systems to demonstrate the superiority of ENIGMA. Specifically, we follow the experimental settings similar to Ghandeharioun et al. [39] and See et al. [40], and show ENIGMA significantly outperforms the existing static evaluation and self-play evaluation methods in both domains.

CHAPTER 2
FINE-TUNING PRE-TRAINED NATURAL LANGUAGE MODELS WITH LIMITED DATA THROUGH REGULARIZED OPTIMIZATION

This chapter focuses on transfer learning with limited data. The content is based on the following publication:

Haoming Jiang et al. (2020b). "SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2177–2190.

2.1 Overview

Transfer learning considers the scenario, where we have limited labeled data from the target domain for a certain task, but we have relevant tasks with a large amount of data from different domains (also known as out-of-domain data). The goal is to transfer the knowledge from the high-resource domains to the low-resource target domain. Here we are particularly interested in the popular two-stage transfer learning framework [5]. The first stage is pre-training, where a high-capacity model is trained for the out-of-domain high-resource relevant tasks. The second stage is fine-tuning, where the high-capacity model is adapted to the low-resource task in the target domain.

Due to the limited data from the target task/domain and the **extremely high complexity** of the pre-trained model, **aggressive fine-tuning** often makes the adapted model overfit the training data of the target task/domain and therefore does not generalize well to unseen data. To mitigate this issue, the fine-tuning methods often rely on hyper-parameter tuning heuristics. For example, Howard and Ruder [19] use a heuristic learning rate schedule and gradually unfreeze the layers of the language model to improve the fine-tune performance;

Peters, Ruder, and Smith [20] give a different suggestion that they only adapt certain layers and freeze the others; [21, 22] propose to add additional layers to the pre-trained model and fine-tune both of them or only the additional layers. However, these methods require significant tuning efforts.

To fully harness the power of fine-tuning in a more principled manner, we propose a new learning framework for robust and efficient fine-tuning on the pre-trained language models through regularized optimization techniques. Specifically, our framework consists of two important ingredients for preventing overfitting:

(I) To effectively control the **extremely high complexity** of the model, we propose a *Smoothness-inducing Adversarial Regularization* technique. Our proposed regularization is motivated by local shift sensitivity in existing literature on robust statistics. Such regularization encourages the output of the model not to change much, when injecting a small perturbation to the input. Therefore, it enforces the smoothness of the model, and effectively controls its capacity [53].

(II) To prevent **aggressive updating**, we propose a class of *Bregman Proximal Point Optimization* methods. Our proposed optimization methods introduce a trust-region-type regularization [54] at each iteration, and then update the model only within a small neighborhood of the previous iterate. Therefore, they can effectively prevent aggressive updating and stabilize the fine-tuning process.

Notation: We use $f(x;\theta)$ to denote a mapping f associated with the parameter θ from input sentences x to an output space, where the output is a multi-dimensional probability simplex for classification tasks and a scalar for regression tasks. $\Pi_{\mathcal{A}}$ denotes the projection operator to the set $\mathcal{A}$. $\mathcal{D}_{KL}(P||Q) = \sum_k p_k \log(p_k/q_k)$ denotes the KL-divergence of two discrete distributions P and Q with the associated parameters of p_k and q_k, respectively.

2.2 The Proposed Method

We describe the proposed learning framework – ***SMART*** for robust and efficient fine-tuning of pre-trained language models. Our framework consists of two important ingredients: ***SM**oothness-inducing **A**dversarial **R**egularization* and *B**R**egman p**R**oximal poin**T** op**T**imization*[1].

2.2.1 Smoothness-Inducing Adversarial Regularization

We propose to impose an explicit regularization to effectively control the model complexity at the fine-tuning stage. Specifically, given the model $f(\cdot;\theta)$ and n data points of the target task denoted by $\{(x_i, y_i)\}_{i=1}^n$, where x_i's denote the embedding of the input sentences obtained from the first embedding layer of the language model and y_i's are the associated labels, our method essentially solves the following optimization for fine-tuning:

$$\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\mathrm{s}} \mathcal{R}_{\mathrm{s}}(\theta), \tag{2.1}$$

where $\mathcal{L}(\theta)$ is the loss function defined as

$$\mathcal{L}(\theta) = \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell(f(x_i;\theta), y_i),$$

and $\ell(\cdot,\cdot)$ is the loss function depending on the target task, $\lambda_{\mathrm{s}} > 0$ is a tuning parameter, and $\mathcal{R}_{\mathrm{s}}(\theta)$ is the smoothness-inducing adversarial regularizer. Here we define $\mathcal{R}_{\mathrm{s}}(\theta)$ as

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n} \sum_{i=1}^n \max_{\|\widetilde{x}_i - x_i\|_p \le \epsilon} \ell_{\mathrm{s}}(f(\widetilde{x}_i;\theta), f(x_i;\theta)),$$

[1]The complete name of our proposed method is ***SMAR**^{3}**T**^{2}*, but we use ***SMART*** for notational simplicity.

where $\epsilon > 0$ is a tuning parameter. Note that for classification tasks, $f(\cdot;\theta)$ outputs a probability simplex and ℓ_{s} is chosen as the symmetrized KL-divergence, i.e.,

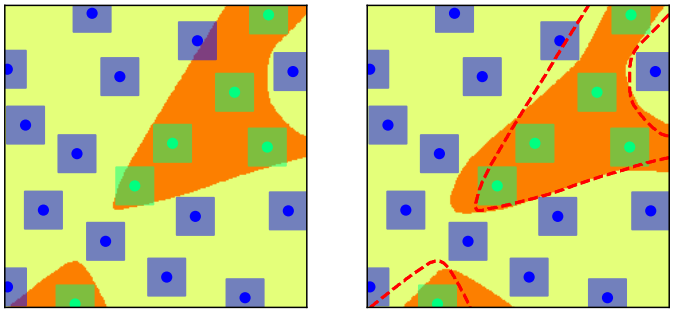
$$\ell_{\mathrm{s}}(P,Q) = \mathcal{D}_{\mathrm{KL}}(P||Q) + \mathcal{D}_{\mathrm{KL}}(Q||P);$$

For regression tasks, $f(\cdot;\theta)$ outputs a scalar and ℓ_{s} is chosen as the squared loss, i.e., $\ell_s(p,q) = (p-q)^2$. Note that the computation of $\mathcal{R}_{\mathrm{s}}(\theta)$ involves a maximization problem and can be solved efficiently by projected gradient ascent.

We remark that the proposed smoothness-inducing adversarial regularizer was first used in Miyato et al. [59] for semi-supervised learning with $p = 2$, and then in Shu et al. [60] for unsupervised domain adaptation with $p = 2$, and more recently in Zhang et al. [61] for harnessing the adversarial examples in image classification with $p = \infty$. To the best of our knowledge, we are the first applying such a regularizer to fine-tuning of pre-trained language models.

The smoothness-inducing adversarial regularizer is essentially measuring the local Lipschitz continuity of f under the metric ℓ_s. More precisely speaking, the output of f does not change much if we inject a small perturbation (ℓ_p norm bounded by ϵ) to x_i. Therefore, by minimizing the objective in (2.1), we can encourage f to be smooth within the neighborhoods of all x_i's. Such a smoothness-inducing property is particularly helpful to prevent overfitting and improve generalization on a low resource target domain for a certain task. An illustration is provided in Figure 2.1.

Note that the idea of measuring the local Lipschitz continuity is similar to the local shift sensitivity criterion in existing literature on robust statistics, which dates back to 1960's [62, 63]. This criterion has been used to characterize the dependence of an estimator on the value of one of the sample points.



(a) (b)

Figure 2.1: Decision boundaries learned without (a) and with (b) smoothness-inducing adversarial regularization, respectively. The red dotted line in (b) represents the decision boundary in (a). As can be seen, the output f in (b) does not change much within the neighborhood of training data points.

2.2.2 Bregman Proximal Point Optimization

We propose to develop a class of Bregman proximal point optimization methods to solve (2.1). Such optimization methods impose a strong penalty at each iteration to prevent the model from aggressive update. Specifically, we use a pre-trained model as the initialization denoted by $f(\cdot;\theta_0)$. At the $(t+1)$-th iteration, the vanilla Bregman proximal point (VBPP) method takes

$$\theta_{t+1} = \text{argmin}_\theta \, \mathcal{F}(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \theta_t), \tag{2.2}$$

where $\mu > 0$ is a tuning parameter, and $\mathcal{D}_{\text{Breg}}(\cdot,\cdot)$ is the Bregman divergence defined as

$$\mathcal{D}_{\text{Breg}}(\theta, \theta_t) = \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell_{\text{s}}(f(x_i;\theta), f(x_i;\theta_t)),$$

where ℓ_s is defined in Section 2.2.1. As can be seen, when μ is large, the Bregman divergence at each iteration of the VBPP method essentially serves as a strong regularizer and prevents θ_{t+1} from deviating too much from the previous iterate θ_t. This is also known as the trust-region type iteration in existing optimization literature [54]. Consequently, the Bregman proximal point method can effectively retain the knowledge of the out-of-domain

data in the pre-trained model $f(\cdot;\theta_0)$. Since each subproblem (2.2) of VBPP does not admit a closed-form solution, we need to solve it using SGD-type algorithms such as ADAM. Note that we do not need to solve each subproblem until convergence. A small number of iterations are sufficient to output a reliable initial solution for solving the next subproblem.

Moreover, the Bregman proximal point method is capable of adapting to the information geometry (See more details in Raskutti and Mukherjee [64]) of machine learning models and achieving better computational performance than the standard proximal point method (i.e., $\mathcal{D}_{\mathrm{Breg}}(\theta,\theta_t)=\|\theta-\theta_t\|_2^2$) in many applications.

Acceleration by Momentum. Similar to other optimization methods in existing literature, we can accelerate the Bregman proximal point method by introducing an additional momentum to the update. Specifically, at the $(t+1)$-th iteration, the momentum Bregman proximal point (MBPP) method takes

$$\theta_{t+1}=\mathrm{argmin}_{\theta}\,\mathcal{F}(\theta)+\mu\mathcal{D}_{\mathrm{Breg}}(\theta,\widetilde{\theta}_t), \tag{2.3}$$

where $\widetilde{\theta}_t=(1-\beta)\theta_t+\beta\widetilde{\theta}_{t-1}$ is the exponential moving average and $\beta\in(0,1)$ is the momentum parameter. The MBPP method is also called the "Mean Teacher" method in existing literature [65] and has been shown to achieve state-of-the-art performance in popular semi-supervised learning benchmarks. For convenience, we summarize the MBPP method in Algorithm 1.

2.3 Experiment – Main Results

We demonstrate the effectiveness of SMART for fine-tuning large language models using GLUE [66] by comparing with existing state-of-the-art methods. Dataset details can be found in Appendix A.1.

Algorithm 1 SMART: We use the smoothness-inducing adversarial regularizer with $p = \infty$ and the momentum Bregman proximal point method.

Notation: For simplicity, we denote $g_i(\widetilde{x}_i, \bar{\theta}_s) = \frac{1}{|\mathcal{B}|}\sum_{x_i \in \mathcal{B}} \nabla_{\widetilde{x}} \ell_{\mathrm{s}}(f(x_i; \bar{\theta}_s), f(\widetilde{x}_i; \bar{\theta}_s))$ and $\mathrm{AdamUpdate}_{\mathcal{B}}$ denotes the ADAM update rule for optimizing (2.3) using the mini-batch $\mathcal{B}$; $\Pi_{\mathcal{A}}$ denotes the projection to $\mathcal{A}$.

Input: T: the total number of iterations, $\mathcal{X}$: the dataset, θ_0: the parameter of the pre-trained model, S: the total number of iteration for solving (2.2), σ^2: the variance of the random initialization for $\widetilde{x}_i$'s, $T_{\widetilde{x}}$: the number of iterations for updating $\widetilde{x}_i$'s, η: the learning rate for updating $\widetilde{x}_i$'s, β: momentum parameter.

$\widetilde{\theta}_1 \leftarrow \theta_0$
for $t = 1, .., T$ **do**
  $\bar{\theta}_1 \leftarrow \theta_{t-1}$
  for $s = 1, .., S$ **do**
    Sample a mini-batch $\mathcal{B}$ from $\mathcal{X}$
    For all $x_i \in \mathcal{B}$, initialize $\widetilde{x}_i \leftarrow x_i + \nu_i$ with $\nu_i \sim \mathcal{N}(0, \sigma^2 I)$
    for $m = 1, .., T_{\widetilde{x}}$ **do**
      $\widetilde{g}_i \leftarrow \frac{g_i(\widetilde{x}_i, \bar{\theta}_s)}{\left\|g_i(\widetilde{x}_i, \bar{\theta}_s)\right\|_\infty}$
      $\widetilde{x}_i \leftarrow \Pi_{\|\widetilde{x}_i - x\|_\infty \leq \epsilon}(\widetilde{x}_i + \eta \widetilde{g}_i)$
    end for
    $\bar{\theta}_{s+1} \leftarrow \mathrm{AdamUpdate}_{\mathcal{B}}(\bar{\theta}_s)$
  end for
  $\theta_t \leftarrow \bar{\theta}_S$
  $\widetilde{\theta}_{t+1} \leftarrow (1 - \beta)\bar{\theta}_S + \beta \widetilde{\theta}_t$
end for

Output: θ_T

2.3.1 Implementation Details

Our implementation of SMART is based on BERT[2] [67], RoBERTa [3] [16], MT-DNN [4] [68] and HNN[5]. We used ADAM [69] and RADAM [70] as our optimizers with a learning rate in the range $\in \{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ and a batch size $\in \{16, 32, 64\}$. The maximum number of epochs was set to 6. A linear learning rate decay schedule with warm-up of 0.1 was used, unless stated otherwise. We also set the dropout rate of all the task specific layers as 0.1, except 0.3 for MNLI and 0.05 for CoLA. To avoid gradient exploding, we clipped the gradient norm within 1. All the texts were tokenized using

[2] https://github.com/huggingface/transformers
[3] https://github.com/pytorch/fairseq
[4] https://github.com/namisan/mt-dnn
[5] https://github.com/namisan/mt-dnn/tree/master/hnn

wordpieces and were chopped to spans no longer than 512 tokens. For SMART, we set the perturbation size $\epsilon = 10^{-5}$ and $\sigma = 10^{-5}$. We set $\mu = 1$ and $\lambda_s \in \{1, 3, 5\}$. The learning rate η in Algorithm 1 is set to 10^{-3}. We set β = 0.99 for the first 10% of the updates ($t \leq 0.1T$) and $\beta = 0.999$ for the rest of the updates ($t > 0.1T$) following [65]. Lastly, we simply set $S = 1, T_{\widetilde{x}} = 1$ in Algorithm 1.

2.3.2 GLUE Main Results

We compare SMART with a range of strong baselines including large pre-trained models and approaches with adversarial training, and a list of state-of-the-art models that have been submitted to the GLUE leaderboard. SMART is a generic framework, we evaluate our framework on two pre-trained models, the $\text{BERT}_{\text{BASE}}$ model [8] and the $\text{RoBERTa}_{\text{LARGE}}$ model [16], which are available publicly. Most of our analyses are done with the $\text{BERT}_{\text{BASE}}$ to make our results comparable to other work, since $\text{BERT}_{\text{BASE}}$ has been widely used as a baseline. To make our result comparable to other state-of-the-art models, we also evaluate the framework on the $\text{RoBERTa}_{\text{LARGE}}$ model.

• BERT [8]: This is the $\text{BERT}_{\text{BASE}}$ model released by the authors. In Devlin et al. [8], authors only reported the development results on a few tasks, thus we reproduced the baseline results, which are denoted by $\textbf{BERT}_{\textbf{ReImp}}$.

• RoBERTa [16]: This is the $\text{RoBERTa}_{\text{LARGE}}$ released by authors, and we present the reported results on the GLUE dev.

• PGD, FreeAT, FreeLB [55]: They are three adversarial training approaches built on top of the $\text{RoBERTa}_{\text{LARGE}}$.

• SMART: our proposed method as described in section 2.2. We use both the $\text{BERT}_{\text{BASE}}$ model ($\text{SMART}_{\text{BERT}}$) and the $\text{RoBERTa}_{\text{LARGE}}$ model ($\text{SMART}_{\text{RoBERTa}}$) as the pretrained model to evaluate the effectiveness of SMART.

The main results are reported in Table 2.1. This table can be clustered into two groups based on different pretrained models: the first group is based on the $\text{BERT}_{\text{BASE}}$ model and

the second group is based on the $\text{RoBERTa}_{\text{LARGE}}$ model. The detailed discussions are as follows.

For a fair comparison, we reproduced the BERT baseline ($\text{BERT}_{\text{ReImp}}$), since several results on the GLUE development set were missed. Our reimplemented BERT baseline is even stronger than the originally reported results in Devlin et al. [8]. For instance, the reimplemented model obtains 84.5% (vs. 84.4%) on MNLI in-domain development in terms of accuracy. On SST-2, $\text{BERT}_{\text{ReImp}}$ outperforms BERT by 0.2% (92.9% vs. 92.7%) accuracy. All these results demonstrate the fairness of our baselines.

Table 2.1: Main results on GLUE development set. The best result on each task produced by a single model is in **bold** and "-" denotes the missed result.

Model	MNLI-m/mm Acc	QQP Acc/F1	RTE Acc	QNLI Acc	MRPC Acc/F1	CoLA Mcc	SST Acc	STS-B P/S Corr
$\text{BERT}_{\text{BASE}}$								
BERT [8]	84.4/-	-	-	88.4	-/86.7	-	92.7	-
$\text{BERT}_{\text{ReImp}}$	84.5/84.4	90.9/88.3	63.5	91.1	84.1/89.0	54.7	92.9	89.2/88.8
$\text{SMART}_{\text{BERT}}$	**85.6/86.0**	**91.5/88.5**	**71.2**	**91.7**	**87.7/91.3**	**59.1**	**93.0**	**90.0/89.4**
$\text{RoBERTa}_{\text{LARGE}}$								
RoBERTa [16]	90.2/-	92.2/-	86.6	94.7	-/90.9	68.0	96.4	92.4/-
PGD [55]	90.5/-	92.5/-	87.4	94.9	-/90.9	69.7	96.4	92.4/-
FreeAT [55]	90.0/-	92.5/-	86.7	94.7	-/90.7	68.8	96.1	92.4/-
FreeLB [55]	90.6/-	**92.6**/-	88.1	95.0	-/91.4	**71.1**	96.7	92.7/-
$\text{SMART}_{\text{RoBERTa}}$	**91.1/91.3**	92.4/89.8	**92.0**	**95.6**	**89.2/92.1**	70.6	**96.9**	**92.8/92.6**

Table 2.2: GLUE test set results scored using the GLUE evaluation server. The state-of-the-art results are in **bold**. All the results were obtained online from https://gluebenchmark.com/leaderboard on December 5, 2019. SMART uses the classification objective on QNLI. Model references: [1] Liu et al. [16]; [2]Zhu et al. [55]; [3]Wang et al. [71]; [4]Lan et al. [14]; [5] Devlin et al. [8]; [6] Liu et al. [18]; [7] Raffel et al. [15] and [8] He et al. [72], Kocijan et al. [73]. * ALBERT uses a model similar in size, architecture and computation cost to a 3,000M BERT (though it has dramatically fewer parameters due to parameter sharing). † Mixed results from ensemble and single of MT-DNN-SMART and with data augmentation.

Model /#Train	CoLA 8.5k	SST 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	**Score**	#param
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1	-
Ensemble Models												
RoBERTa[1]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5	356M
FreeLB[2]	68.0	96.8	93.1/90.8	92.4/92.2	**74.8**/90.3	91.1/90.7	98.8	88.7	89.0	50.1	88.8	356M
ALICE[3]	69.2	97.1	93.6/91.5	92.7/92.3	74.4/**90.7**	90.7/90.2	**99.2**	87.3	89.7	47.8	89.0	340M
ALBERT[4]	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3/91.0	**99.2**	89.2	91.8	50.2	89.4	235M*
MT-DNN-SMART†	69.5	**97.5**	**93.7/91.6**	**92.9/92.5**	73.9/90.2	91.0/90.8	**99.2**	89.7	94.5	50.2	**89.9**	356M
Single Model												
$\text{BERT}_{\text{LARGE}}$[5]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5	335M
MT-DNN[6]	62.5	95.6	90.0/86.7	88.3/87.7	72.4/89.6	86.7/86.0	93.1	75.5	65.1	40.3	82.7	335M
T5[8]	**70.8**	97.1	91.9/89.2	92.5/92.1	74.6/90.4	**92.0/91.7**	96.7	**92.5**	**93.2**	**53.1**	89.7	11,000M
$\text{SMART}_{\text{RoBERTa}}$	65.1	**97.5**	**93.7/91.6**	**92.9/92.5**	74.0/90.1	91.0/90.8	95.4	87.9	91.8[8]	50.2	88.4	356M

Comparing with two strong baselines BERT and RoBERTa [6], our proposed SMART, including $\text{SMART}_{\text{BERT}}$ and $\text{SMART}_{\text{RoBERTa}}$, consistently outperforms them across all 8 GLUE tasks by a big margin. Comparing with BERT, $\text{SMART}_{\text{BERT}}$ obtained 85.6% (vs. 84.5%) and 86.0% (vs. 84.4%) in terms of accuracy, which is 1.1% and 1.6% absolute improvement, on the MNLI in-domain and out-domain settings. Even comparing with the state-of-the-art model RoBERTa, $\text{SMART}_{\text{RoBERTa}}$ improves 0.8% (91.1% vs. 90.2%) on MNLI in-domain development set. Interestingly, on the MNLI task, the performance of SMART on the out-domain setting is better than the in-domain setting, e.g., (86.0% vs. 85.6%) by $\text{SMART}_{\text{BERT}}$ and (91.3% vs. 91.1%) by $\text{SMART}_{\text{RoBERTa}}$, showing that our proposed approach alleviates the domain shifting issue. Furthermore, on the small tasks, the improvement of SMART is even larger. For example, comparing with BERT, $\text{SMART}_{\text{BERT}}$ obtains 71.2% (vs. 63.5%) on RTE and 59.1% (vs. 54.7%) on CoLA in terms of accuracy, which are 7.7% and 4.4% absolute improvement for RTE and CoLA, respectively; similarly, $\text{SMART}_{\text{RoBERTa}}$ outperforms RoBERTa 5.4% (92.0% vs. 86.6%) on RTE and 2.6% (70.6% vs. 68.0%) on CoLA.

We also compare SMART with a range of models which used adversarial training such as FreeLB. From the bottom rows in Table 2.1, SMART outperforms PGD and FreeAT across the all 8 GLUE tasks. Comparing with the current state-of-the-art adversarial training model, FreeLB, SMART outperforms it on 6 GLUE tasks out of a total of 8 tasks (MNLI, RTE, QNLI, MRPC, SST-2 and STS-B) showing the effectiveness of our model.

Table 2.2 summarizes the current state-of-the-art models on the GLUE leaderboard. SMART obtains a competitive result comparing with T5 [15], which is the leading model at the GLUE leaderboard. T5 has 11 billion parameters, while SMART only has 356 millions. Among this super large model (T5) and other ensemble models (e.g., ALBERT, ALICE), SMART, which is a single model, still sets new state-of-the-art results on SST-2, MRPC and STS-B. By combining with the Multi-task Learning framework (MT-DNN), MT-DNN-

[6] In our experiments, we use BERT referring the $\text{BERT}_{\text{BASE}}$ model, which has 110 million parameters, and RoBERTa referring the $\text{RoBERTa}_{\text{LARGE}}$ model, which has 356 million parameters, unless stated otherwise.

SMART obtains new state-of-the-art on GLUE, pushing the GLUE benchmark to 89.9%. More discussion will be provided in Section 2.4.3.

2.4 Experiment – Analysis and Extension

In this section, we first analyze the effectiveness of each component of the proposed method. We also study that whether the proposed method is complimentary to multi-task learning. We further extend SMART to domain adaptation and use both SNLI [74] and SciTail [75] to evaluate the effectiveness. Finally, we verified the robustness of the proposed method on ANLI [76].

2.4.1 Ablation Study

Note that due to the limitation of time and computational resources, all the experiments reported below are based on the **BERT$_{\text{BASE}}$** model. In this section, we study the importance of each component of SMART: smoothness-inducing adversarial regularization and Bregman proximal point optimization. All models in this study used the BERT$_{\text{BASE}}$ as the encoder for fast training. Furthermore, we also include the BERT$_{\text{BASE}}$ model as an additional baseline for a fair comparison. SMART denotes the proposed model. Then we set λ_s to 0, which denotes as -$\mathcal{R}_{\text{s}}$. The model with $\mu = 0$ is noted as -$\mathcal{D}_{\text{Breg}}$.

Table 2.3: Ablation study of SMART on 5 GLUE tasks. Note that all models used the BERT$_{\text{BASE}}$ model as their encoder.

Model	MNLI Acc	RTE Acc	QNLI Acc	SST Acc	MRPC Acc
BERT	84.5	63.5	91.1	92.9	89.0
SMART	**85.6**	**71.2**	**91.7**	**93.0**	**91.3**
-$\mathcal{R}_{\text{s}}$	84.8	70.8	91.3	92.8	90.8
-$\mathcal{D}_{\text{Breg}}$	85.4	**71.2**	91.6	92.9	91.2

The results are reported in Table 2.3. It is expected that the removal of either component (smooth regularization or proximal point method) in SMART would result in a performance drop. For example, on MNLI, removing smooth regularization leads to a 0.8% (85.6% vs.

84.8) performance drop, while removing the Breg proximal point optimization, results in a performance drop of 0.2% (85.6% vs. 85.4%). It demonstrates that these two components complement each other. Interestingly, all three proposed models outperform the BERT baseline model demonstrating the effectiveness of each module. Moreover, we obersere that the generalization performance benefits more from SMART on small datasets (i.e., RTE and MRPC) by preventing overfitting.

2.4.2 Error Analysis

To understand why SMART improves the performance, we analyze it on the ambiguous samples of MNLI dev set containing 3 classes, where each sample has 5 annotations. Based on the degree of agreement between these annotations, we divide the samples into 4 categories: 1) **5/0/0** all five annotations are the same; 2) **4/1/0** four annotations are the same; 3) **3/2/0** three annotations are the same and the other two annotations are the same; 4) **3/1/1** three annotations are the same and the other two annotations are different.

Figure 2.2 summarizes the results in terms of both accuracy and KL-divergence. The KL-divergence captures the distribution mismatch: $-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{3} p_j(x_i)\log(f_j(x_i))$. For a given sample x_i, the KL-Divergence evaluates the similarity between the model prediction $\{f_j(x_i)\}_{j=1}^{3}$ and the annotation distribution $\{p_j(x_i)\}_{j=1}^{3}$. We observe that $\text{SMART}_{\text{RoBERTa}}$ outperforms RoBERTa across all the settings. Further, on high degree of ambiguity (low degree of agreement), $\text{SMART}_{\text{RoBERTa}}$ obtains an even larger improvement showing its robustness to ambiguity.

2.4.3 SMART with Multi-task Learning

It has been shown that multi-task learning (MTL, Caruana [77] and Liu et al. [78, 18]) has a regularization effect via alleviating overfitting to a specific task. One question is whether MTL helps SMART as well. In this section, we are going to answer this question. Following Liu et al. [18], we first "pre-trained" shared embeddings using MTL with SMART,

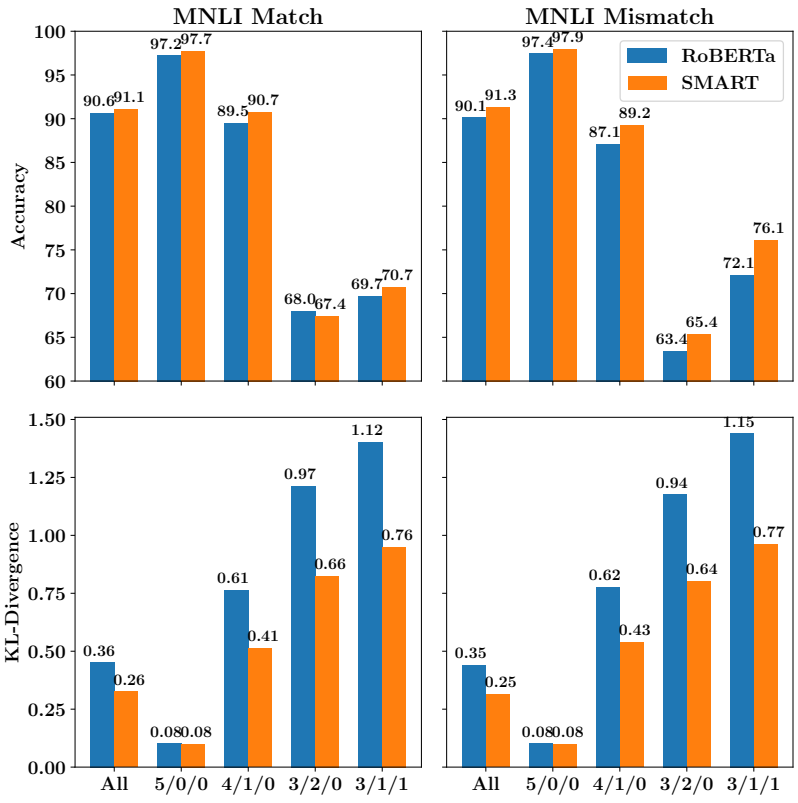


Figure 2.2: Score breakdown by degree of agreement.

denoted as **MT-DNN-SMART** [7], and then adapted the training data on each task on top of the shared embeddings. We also include a baseline which fine-tuned each task on the publicly released MT-DNN checkpoint [8], which is indicated as **MT-DNN-SMART$_{v0}$**.

Table 2.4: Comparison between SMART and MTL.

Model	MNLI Acc	RTE Acc	QNLI Acc	SST Acc	MRPC F1
BERT	84.5	63.5	91.1	92.9	89.0
MT-DNN	85.3	79.1	91.5	**93.6**	89.2
SMART	85.6	71.2	91.6	93.0	91.3
MT-DNN-SMART$_{v0}$	**85.7**	80.2	**92.0**	93.3	91.5
MT-DNN-SMART	**85.7**	**81.2**	**92.0**	93.5	**91.7**

We observe that both MT-DNN and SMART consistently outperform the BERT model

[7] Due to limitation of computational resources, we only trained jointly using MTL on MNLI, RTE, QNLI, SST and MRPC, while MT-DNN was trained on the whole GLUE tasks except CoLA.

[8] It is from: https://github.com/namisan/mt-dnn. Note that we did not use the complicated answer module, e.g., SAN [79].

Table 2.5: Domain adaptation on SNLI and SciTail.

Model	0.1%	1%	10%	100%
SNLI Dataset (Dev Accuracy%)				
#Training Data	549	5,493	54,936	549,367
BERT	52.5	78.1	86.7	91.0
MT-DNN	82.1	85.2	88.4	91.5
MT-DNN-SMART	**82.7**	**86.0**	**88.7**	**91.6**
SciTail Dataset (Dev Accuracy%)				
#Training Data	23	235	2,359	23,596
BERT	51.2	82.2	90.5	94.3
MT-DNN	81.9	88.3	91.1	95.8
MT-DNN-SMART	**82.3**	**88.6**	**91.3**	**96.1**

on all five GLUE tasks. Furthermore, SMART outperforms MT-DNN on MNLI, QNLI, and MRPC, while it obtains worse results on RTE and SST, showing that MT-DNN is a strong counterpart for SMART. By combining these two models, MT-DNN-SMART$_{v0}$ enjoys advantages of both and thus improved the final results. For example, it achieves 85.7% (+0.1%) on MNLI and 80.2% (+1.1%) on RTE comparing with the best results of MT-DNN and SMART demonstrating that these two techniques are orthogonal. Lastly we also trained SMART jointly and then finetuned on each task like Liu et al. [18]. We observe that MT-DNN-SMART outperformes MT-DNN-SMART$_{v0}$ and MT-DNN across all 5 tasks (except MT-DNN on SST) showing that SMART improves the generalization of MTL.

2.4.4 Domain Adaptation

In this section, we evaluate our model on the domain adaptation setting. Following Liu et al. [18], we start with the default training/dev/test set of SNLI and SciTail. Then, we randomly sample 0.1%, 1%, 10% and 100% of its training data, which is used to train a model.

The results are reported in Table 2.5. We observe that both MT-DNN and MT-DNN-SMART significantly outperform the BERT baseline. Comparing with MT-DNN, MT-

DNN-SMART also achieves some improvements indicating the robustness of SMART. Furthermore, MT-DNN-SMART outperforms current state-of-the-art on the SNLI/SciTail test.

Table 2.6: Experiment Result for Each Round of ANLI.

Method	Dev				Test			
	R1	R2	R3	All	R1	R2	R3	All
MNLI + SNLI + ANLI + FEVER								
$\text{BERT}_{\text{LARGE}}$ [76]	-	-	-	-	57.4	48.3	43.5	49.3
$\text{XLNet}_{\text{LARGE}}$ [76]	-	-	-	-	67.6	50.7	48.3	55.1
$\text{RoBERTa}_{\text{LARGE}}$ [76]	-	-	-	-	73.8	48.9	44.4	53.7
$\text{SMART}_{\text{RoBERTa-LARGE}}$	74.5	50.9	47.6	**57.1**	72.4	49.8	50.3	**57.1**
ANLI								
$\text{RoBERTa}_{\text{LARGE}}$ [76]	-	-	-	-	71.3	43.3	43.0	51.9
$\text{SMART}_{\text{RoBERTa-LARGE}}$	74.2	49.5	49.2	**57.1**	72.4	50.3	49.5	**56.9**

2.4.5 Results on SNLI and SciTail

In Table 2.7, we compare our methods, using all in-domain training data, against several state-of-the-art models. We observe that SMART obtains the same improvement on SNLI in the BERT setting. Combining SMART with MT-DNN achieves a significant improvement, e.g., our BASE model even outperforms the $\text{BERT}_{\text{LARGE}}$ model. Similar observation is found on SciTail and in the $\text{BERT}_{\text{LARGE}}$ model setting. We see that incorporating SMART into MT-DNN achieves new state-of-the-art results on both SNLI and SciTail, pushing benchmarks to 91.7% on SNLI and 95.2% on SciTail.

2.4.6 Robustness

One important property of the machine learning model is its robustness to adversarial attack. We test our model on an adversarial natural language inference (ANLI) dataset [76].

We evaluate the performance of SMART on each subset (i.e., R1,R2,R3) of ANLI dev and test set. The results are presented in Table 2.6. Table 2.6 shows the results of training on combined NLI data (ANLI [76] + MNLI [82] + SNLI [74] + FEVER [83]) and training

Table 2.7: Results on the SNLI and SciTail dataset.

Model	Dev	Test
SNLI Dataset (Accuracy%)		
$\text{BERT}_{\text{BASE}}$	91.0	90.8
$\text{BERT}_{\text{BASE}}$+SRL[80]	-	90.3
$\text{MT-DNN}_{\text{BASE}}$	91.4	91.1
$\text{SMART}_{\text{BERT-BASE}}$	91.4	91.1
$\text{MT-DNN-SMART}_{\text{BASEv0}}$	91.7	91.4
$\text{MT-DNN-SMART}_{\text{BASE}}$	91.7	91.5
$\text{BERT}_{\text{LARGE}}$+SRL[80]	-	91.3
$\text{BERT}_{\text{LARGE}}$	91.7	91.0
$\text{MT-DNN}_{\text{LARGE}}$	92.2	91.6
$\text{MT-DNN-SMART}_{\text{LARGEv0}}$	**92.6**	**91.7**
SciTail Dataset (Accuracy%)		
GPT [81]	-	88.3
$\text{BERT}_{\text{BASE}}$	94.3	92.0
$\text{MT-DNN}_{\text{BASE}}$	95.8	94.1
$\text{SMART}_{\text{BERT-BASE}}$	94.8	93.2
$\text{MT-DNN-SMART}_{\text{BASEv0}}$	96.0	94.0
$\text{MT-DNN-SMART}_{\text{BASE}}$	96.1	94.2
$\text{BERT}_{\text{LARGE}}$	95.7	94.4
$\text{MT-DNN}_{\text{LARGE}}$	96.3	95.0
$\text{SMART}_{\text{BERT-LARGE}}$	96.2	94.7
$\text{MT-DNN-SMART}_{\text{LARGEv0}}$	**96.6**	**95.2**

on only ANLI data. In the combined data setting, we obverse that $\text{SMART}_{\text{RoBERTa-LARGE}}$ obtains the best performance compared with all the strong baselines, pushing benchmarks to 57.1%. In case of the $\text{RoBERTa}_{\text{LARGE}}$ baseline, $\text{SMART}_{\text{RoBERTa-LARGE}}$ outperforms 3.4% absolute improvement on dev and 7.4% absolute improvement on test, indicating the robustness of SMART. We obverse that in the ANLI-only setting, $\text{SMART}_{\text{RoBERTa-LARGE}}$ outperforms the strong $\text{RoBERTa}_{\text{LARGE}}$ baseline with a large margin, +5.2% (57.1% vs. 51.9%)

2.5 Discussion and Related Work

Our proposed regularization technique is related to several existing works [59, 61, 60]. These works consider similar regularization techniques, but target at other applications with different motivations, e.g., semi-supervised learning, unsupervised domain adaptation and harnessing adversarial examples in image classification.

Our proposed optimization technique covers a large class of Bregman proximal point methods in existing literature on optimization, including vanilla proximal point method proposed in Rockafellar [84], generalized proximal point method [85, 86], accelerated proximal point method, and other variants [87, 88, 89].

There is a related fine-tuning method – FreeLB Zhu et al. [55], which adapted a robust adversarial training method. However, our framework focuses on the local smoothness, leading to a significant performance improvement. More discussion and comparison are provided in Section 2.3.

2.6 Conclusion

We propose a robust and efficient computation framework, SMART, for fine-tuning large scale pre-trained natural language models in a principled manner. The framework effectively alleviates the overfitting and aggressive updating issues in the fine-tuning stage. SMART includes two important ingredients: 1) smooth-inducing adversarial regularization; 2) Bregman proximal point optimization. Our empirical results suggest that SMART improves the performance on many NLP benchmarks (e.g., GLUE, SNLI, SciTail, ANLI) with the state-of-the-art pre-trained models (e.g., BERT, MT-DNN, RoBERTa). We also demonstrate that the proposed framework is applicable to domain adaptation and results in a significant performance improvement. Our proposed fine-tuning framework can be generalized to solve other transfer learning problems. We will explore this direction as future work.

CHAPTER 3
WEAKLY SUPERVISED NAMED ENTITY RECOGNITION WITH TRANSFER LEARNING AND SELF-TRAINING

This chapter focuses on named entity recognition with weak supervision. The content is based on the following publications:

Chen Liang et al. (2020). "Bond: Bert-assisted open-domain named entity recognition with distant supervision". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1054–1064

Haoming Jiang et al. (2020a). "Named Entity Recognition with Small Strongly Labeled and Large Weakly Labeled Data". In: *Preprint*

We remark that the proposed self-training method can also be extended to NLP tasks other than named entity recognition, which will be discussed in Chapter 4.

3.1 Overview

Named Entity Recognition (NER) is the task of detecting mentions of real-world entities from text and classifying them into predefined types (e.g., locations, persons, organizations). It is a core task in knowledge extraction and is important to various downstream applications such as user interest modeling [91], question answering [92] and dialogue systems [93]. As NER tasks require token-level labels, annotating a large number of documents can be expensive, time-consuming, and prone to human errors. In many real-life scenarios, the lack of labeled data has become the biggest bottleneck that prevents deep learning models from being adopted for NER tasks.

To tackle the label scarcity issue, one approach is to use weak/distant supervision to generate labels automatically. In weak supervision, the labeling procedure is to match

the tokens in the target corpus with concepts in knowledge bases (e.g. Wikipedia[1] and YAGO[2]), which are usually easy and cheap to access. Nevertheless, the labels generated by the matching procedure suffer from two major challenges. The first challenge is *incomplete annotation*, which is caused by the limited coverage of existing knowledge bases. Take two common open-domain NER datasets as examples. From Table 3.1, we find that the coverage of tokens on both datasets is very low (less than 60%).This issue renders many entities mentions unmatched and produces many false-positive labels, which can hurt subsequent NER model training significantly. The second challenge is *noisy annotation*. The annotation is often noisy due to the labeling ambiguity – the same entity mention can be mapped to multiple entity types in the knowledge bases. For instance, the entity mention '*Liverpool*' can be mapped to both '*Liverpool City*' (type: `LOC`) and '*Liverpool Football Club*' (type: `ORG`) in the knowledge base. While existing methods adopt label induction methods based on type popularity, they will potentially lead to a matching bias toward popular types. Consequently, it can lead to many false-positive samples and hurt the performance of NER models. What's worse, there is often a trade-off between the label accuracy and coverage: generating the high-quality label requires setting strict matching rules which may not generalize well for all the tokens and thus reduce the coverage and introduce false-negative labels. On the other hand, increasing the coverage of annotation suffers from the increasing number of incorrect labels due to label ambiguity. From the above, it is still very challenging to generate high-quality labels with high coverage to the target corpus.

Several studies have attempted to address the above challenges in weakly-supervised NER. To address the label incompleteness issue, some works adopt the partial annotation CRFs to consider all possible labels for unlabeled tokens [94, 95], but they still require a considerable amount of annotated tokens or external tools. To address the label noise is-

[1]https://www.wikipedia.org/

[2]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

sue, Ni, Dinu, and Florian [96] use heuristic rules to filter out sentences with low matching quality. However, this filtering strategy improves the precision at the expense of lowering the recall. Cao et al. [97] attempt to induce labels for entity mentions based on their occurrence popularity in the concept taxonomy, which can suffer from labeling bias and produce mislabeled data. Moreover, most of the methods mainly focus on NER tasks in specific domains (e.g. biomedical, chemistry, etc.) where the ambiguity of the named entity is very low. When the matching ambiguity issue is more severe, such methods will be less effective especially under open-domain scenarios. Till now, training *open-domain* NER models with weak supervision remains a challenging problem.

We propose our model BOND, short for **B**ERT-Assisted **O**pen-Domain **N**amed entity recognition with **D**istant Supervision, which learns accurate named entity taggers from weak supervision without any restriction on the domain or the content of the corpora. To address the challenges in learning from weak supervision, our approach leverages the power of pre-trained language models (e.g., ELMo [6], BERT [8], XLnet [11]) which are particularly attractive to this task due to the following merits: *First*, they are very large neural networks trained with huge amounts of unlabeled data in a *completely unsupervised manner*, which can be cheaply obtained; *Second*, due to their massive sizes (usually having hundreds of millions or billions of parameters), they have *strong expressive power* to capture general semantics and syntactic information effectively. These language models have achieved state-of-the-art performance in many popular NLP benchmarks with appropriate fine-tuning [8, 16, 11, 98, 99], which demonstrates their strong ability in modeling the text data.

To fully harness the power of pre-trained language models for tackling the two challenges, we propose a two-stage training framework. In the first stage, we fine-tune the RoBERTa model [16] with weakly-matched labels to essentially transfer the semantic knowledge in RoBERTa, which will improve the quality of prediction induced from weak supervision. It is worth noting that we adopt early stopping to prevent the model from

overfitting to the incomplete annotated labels[3] and significantly improve the recall. Then we use the RoBERTa model to predict a set of pseudo soft-labels for all data. In the second stage, we replace the weakly-matched labels with the pseudo soft-labels and design a *teacher-student* framework to further improve the recall. The *student* model is first initialized by the model learned in the first stage and trained using pseudo soft-labels. Then, we update the *teacher* model from the *student* model in the previous iteration to generate a new set of pseudo-labels for the next iteration to continue the training of the *student* model. This *teacher-student* framework enjoys the merit that it progressively improves the model confidence over data. In addition, we select samples based on the prediction confidence of the *student* model to further improve the quality of soft labels. In this way, we can better exploit both the knowledge base information and the language models and improve the model fitting.

We conduct comprehensive experiments on 5 datasets for named entity recognition tasks with weak supervision. Our proposed method significantly outperforms state-of-the-art weakly supervised NER competitors in all 5 datasets (4 of which by significant margins).

3.2 Preliminaries

We briefly introduce the weakly-supervised NER problem and the pre-trained language models.

3.2.1 Weakly Supervised NER

NER is the process of locating and classifying named entities in text into predefined entity categories, such as person names, organizations, locations, etc. Formally, given a sentence with N tokens $\boldsymbol{X} = [x_1, ..., x_N]$, an entity is a span of tokens $\boldsymbol{s} = [x_i, ..., x_j]$ $(0 \leq i \leq j \leq N)$ associated with an entity type. Based on the `BIO` schema [100], NER is typically formulated as a sequence labeling task of assigning a sequence of labels $\boldsymbol{Y} = [y_1, ..., y_N]$

[3]Here the incomplete annotated labels refer to tokens wrongly labeled as type '`O`'.

to the sentence $\boldsymbol{X}$. Specifically, the first token of an entity mention with type `X` is labeled as `B-X`; the other tokens inside that entity mention are labeled as `I-X`; and the non-entity tokens are labeled as `O`.

For (fully) supervised NER, we are given M sentences that are already annotated at token level, denoted as $\{(\boldsymbol{X}_m, \boldsymbol{Y}_m)\}_{m=1}^{M}$. Let $f(\boldsymbol{X}; \theta)$ denote an NER model, which can compute N probability simplexes for predicting the entity labels of any new sentence $\boldsymbol{X}$, where θ is the parameter of the NER model. We train such a model by minimizing the following loss over $\{(\boldsymbol{X}_m, \boldsymbol{Y}_m)\}_{m=1}^{M}$:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{\mathrm{M}} \ell(\boldsymbol{Y}_m, f(\boldsymbol{X}_m; \theta)), \tag{3.1}$$

where $\ell(\cdot, \cdot)$ is the cross-entropy loss.

For weakly-supervised NER, we do not have access to well-annotated true labels, but only *weak labels* generated by matching unlabeled sentences with external gazetteers or knowledge bases (KBs). The matching can be achieved by string matching [101], regular expressions [102] or heuristic rules (e.g., POS tag constraints). Accordingly, we learn an NER model by minimizing Eq. (3.1) with $\{\boldsymbol{Y}_m\}_{m=1}^{M}$ replaced by their weakly labeled counterparts.

Challenges. The labels generated by weak supervision are often noisy and incomplete. This is particularly true for open-domain NER where there is no restriction on the domain or the content of the corpora. Fries et al. [102] and Giannakopoulos et al. [101] have proposed weakly-supervised NER methods for specific domains (e.g.,, biomedical domain), where the adopted domain-specific gazetteers or KBs are often of high matching quality and yield high precision and high recall weak labels. For the open domain, however, the quality of the weak labels is much worse, as there is more ambiguity and limited coverage over entity types in open-domain KBs. Table 3.1 illustrates the matching quality of weak labels on the open-domain and the biomedical-domain datasets. As can be seen, the weak labels for

Table 3.1: Existing Gazetteer Matching Performance on Open-Domain [103, 104] and Biomedical Domain NER Datasets [95].

Metric	Open-Domain		Biomedical Domains	
	CoNLL03	Tweet	BC5CDR	NCBI-Disease
Entity Types	4	10	2	1
F-1	59.61	35.83	71.98	69.32
Precision	71.91	40.34	93.93	90.59
Recall	50.90	32.22	58.35	56.15

the open-domain datasets suffer from much lower precision and recall. This imposes great challenges to training accurate NER models.

3.2.2 Pre-trained Language Model

Pre-trained language models, such as BERT and its variants (e.g.,, RoBERTa [16], ALBERT [98] and T5 [99]), have achieved state-of-the-art performance in many natural language understanding tasks [105]. These models are essentially massive neural networks based on bi-directional transformer architectures, and are trained using open-domain data in a completely unsupervised manner. The stacked self-attention modules of the transformer architectures can capture deep contextual information, and their non-recurrent structures enable the training to scale to large amounts of open-domain data. For example, the popular BERT-base model contains 110 million parameters, and is trained using the BooksCorpus [106] (800 million words) and English Wikipedia (2500 million words). More importantly, many pre-trained language models have been publicly available online. One does not need to train them from scratch. When applying pre-trained language models to downstream tasks, one only needs to slightly modify the model and adapt the model through efficient and scalable stochastic gradient-type algorithms.

3.3 Two-Stage Framework: BOND

We introduce our proposed two-stage framework–BOND. In the first stage of BOND, we adapt the BERT model to the weakly supervised NER task. In the second stage, we use a

self-training approach to improve the model fitting to the training data. We summarize the BOND framework in Figure 3.1.

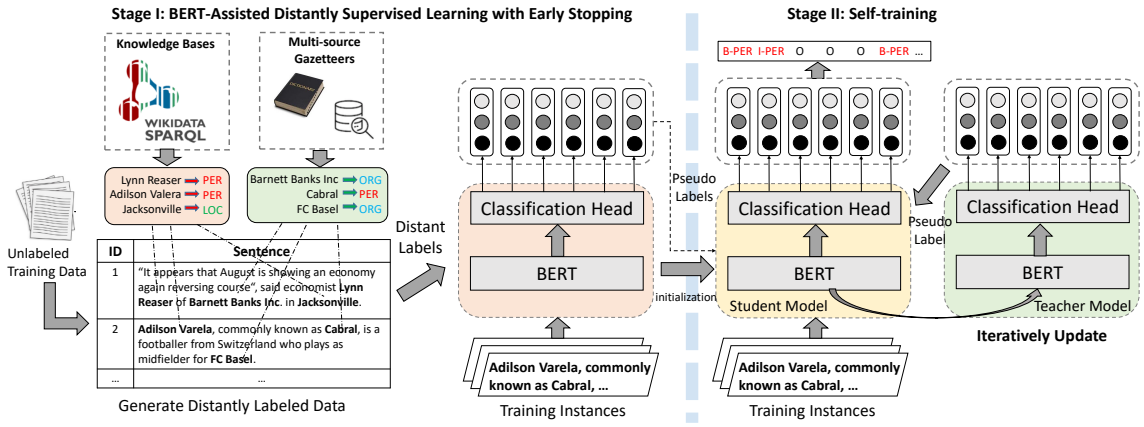


Figure 3.1: The two-stage BOND framework. In Stage I, the pre-trained BERT is adapted to the weakly supervised NER task with early stopping. In Stage II, a student model and a teacher model are first initialized from the model learned in Stage I. Then the student model is trained using pseudo-labels generated by the teacher model. Meanwhile, the teacher model is iteratively updated by the early-stopped student.

3.3.1 Stage I: BERT-Assisted Weakly Supervised Learning with Early Stopping

Before proceeding with our proposed method, we briefly introduce how we generate weak labels for open-domain NER tasks. Our label generation scheme contains two steps: We first identify potential entities by POS tagging and hand-crafted rules. We then query from Wikidata to identify the types of these entities using SPARQL [107] as illustrated in Figure 3.2. We next collect gazetteers from multiple online resources to match more entities in the data [103]. Please refer to the Appendix B for more technical details.

We then proceed with our proposed method. We use $f(\cdot;\theta)$ to denote the NER model parameterized by θ, $f_{n,c}(\cdot;\cdot)$ to denote the probability of the n-th token belonging to the c-th class, and $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^{M}$ to denote the weakly labeled data, where $\boldsymbol{D}_m = [d_{m,1}, ..., d_{m,N}]$ and $\boldsymbol{X}_m = [x_{m,1}, ..., x_{m,N}]$. The NER model $f(\cdot;\theta)$ is learned by mini-

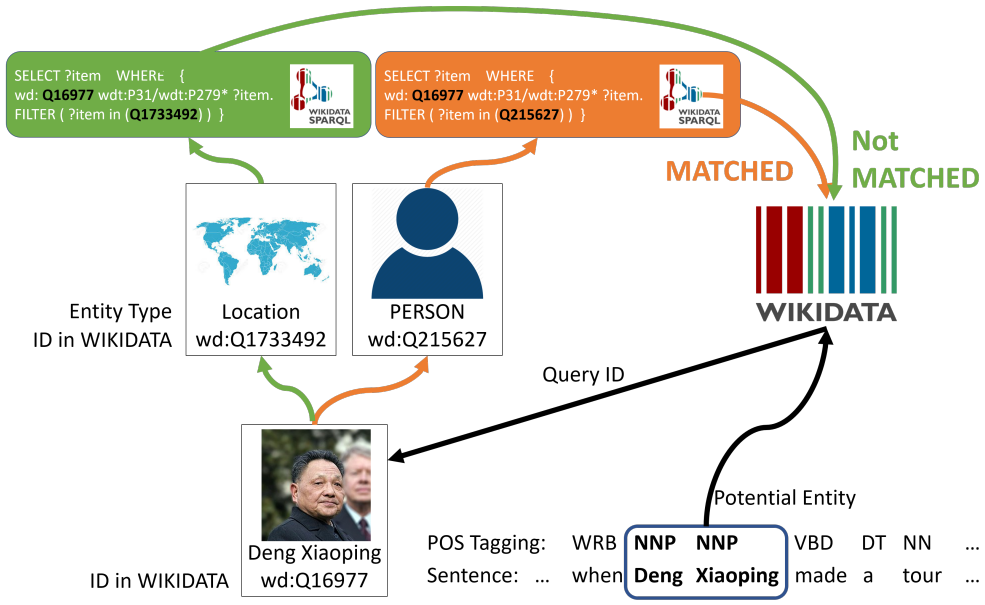


Figure 3.2: Illustration of matching entities from Wikidata

mizing the loss over $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^M$:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \ell(\boldsymbol{D}_m, f(\boldsymbol{X}_m; \theta)), \tag{3.2}$$

where $\ell(\boldsymbol{D}_m, f(\boldsymbol{X}_m; \theta)) = \frac{1}{N} \sum_{n=1}^{N} - \log f_{n,d_{m,n}}(\boldsymbol{X}_m; \theta)$.

The architecture of the NER model $f(\cdot, \cdot)$ is a token-wise NER classifier on top of a pre-trained BERT, as shown in Figure 3.3. The NER classifier takes in the token-wise output embeddings from the pre-trained BERT layers, and gives the prediction on the type for each token. The pre-trained BERT contains rich semantic and syntax knowledge, and yields high quality output embeddings. Using such embeddings as the initialization, we can efficiently adapt the pre-trained BERT to the target NER task using stochastic gradient-type algorithms, e.g., ADAM [69, 108]. Following [99], our adaptation process updates the entire model including both the NER classification layer and the pre-trained BERT layers.

Figure 3.4 illustrates how the pre-trained BERT embeddings help the model adapt to weakly supervised NER tasks. We highlight that BERT is pre-trained through a masked language model (MLM) task, and is capable of predicting the missing words using the

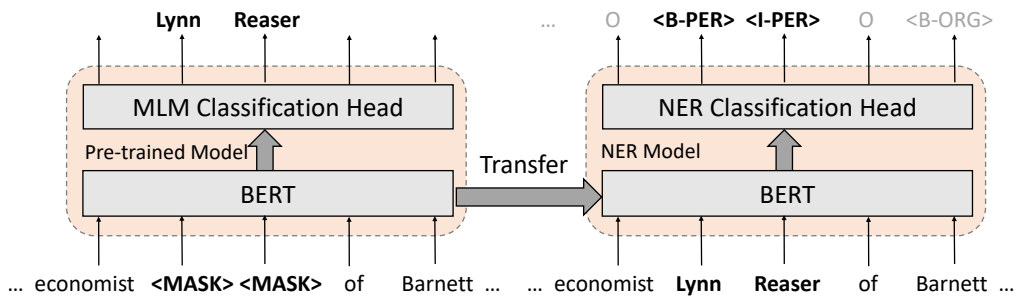


Figure 3.3: Pre-trained Mask Language Model vs. NER Model

Algorithm 2 Stage I: BERT-Assisted Weakly Supervised Learning with Early Stopping

Input: M unlabeled sentences, $\{\boldsymbol{X}_m\}_{m=1}^M$; External KBs including Wikidata and multi-source gazetteers; The NER model with pre-trained BERT layers $f(\cdot;\theta^{(0)})$; The early stopping time T_1; The updating formula of ADAM $\mathcal{T}$.

// Weak Label Generation (DLG)

$$\{\boldsymbol{D}_m\}_{m=1}^M = \text{Matching}(\{\boldsymbol{X}_m, \boldsymbol{D}_m\}_{m=1}^M; \text{External KBs})$$

// Model Adaptation

for $t = 1, 2, ..., T_1$ **do**

  Sample a minibatch $\mathcal{B}_t$ from $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^M$.

  Update the model using ADAM:

$$\theta^{(t)} = \mathcal{T}(\theta^{(t-1)}, \mathcal{B}_t).$$

end for

Output: The early stopped model: $\widehat{\theta} = \theta^{(T_1)}$

contextual information. Such a MLM task shares a lot of similarity with the NER task. Both of them are token-wise classification problems and heavily rely on the contextual information (see Figure 3.3). This naturally enables the semantic knowledge of the pre-trained BERT to be transferred to the NER task. Therefore, the resulting model can better predict the entity types than those trained from scratch using only the weakly labeled data.

Early Stopping. One important strategy we use in the adaptation process is early stopping. Due to the large model capacity as well as the limited and noisy supervision (weak labels), our NER model can overfit the noise in weak labels and forget the knowledge of the pre-trained BERT if without any intervention. Early stopping essentially serves as a strong regularization to prevent such overfitting and improves generalization ability to unseen data.

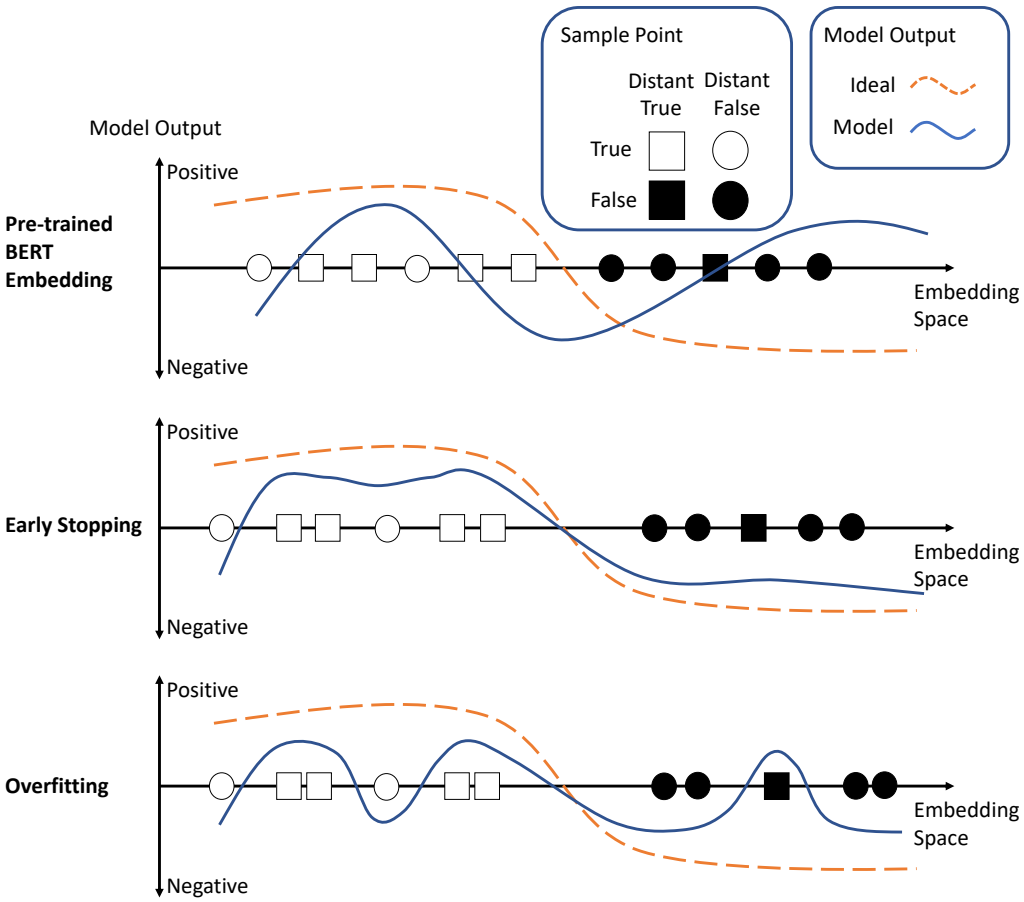


Figure 3.4: Illustration of Stage I. Top) The pre-trained semantic knowledge is transferred to the NER task; Middle) Early stopping leverages the pre-trained knowledge and yields better prediction; Bottom) Without early stopping, the model overfits the noise. The token embeddings are evolving, as we update the pre-trained BERT layers.

Remark 1. Stage I addresses both of the two major challenges in weakly supervised NER tasks: noisy annotation and incomplete annotation. As the semantic knowledge in the pre-trained BERT is transferred to the NER model, the noise is suppressed such that the prediction precision is improved. Moreover, early stopping prevents the model from overfitting the incomplete annotated labels and further improves the recall.

3.3.2 Stage II: Self-Training

We first describe a teacher-student framework of self-training to improve the model fitting, and then we propose to use high-confidence soft labels to further improve the self-training.

The Teacher-student Framework

We use $f(\cdot;\theta_{\text{tea}})$ and $f(\cdot;\theta_{\text{stu}})$ to denote teacher and student models, respectively. Given the model learned in Stage I, $f(\cdot;\widehat{\theta})$, one option is to initialize the teacher model and the student model as:

$$\theta_{\text{tea}}^{(0)} = \theta_{\text{stu}}^{(0)} = \widehat{\theta},$$

and another option is

$$\theta_{\text{tea}}^{(0)} = \widehat{\theta} \quad \text{and} \quad \theta_{\text{stu}}^{(0)} = \theta_{\text{BERT}}, \tag{3.3}$$

where θ_{BERT} denotes the initial model with the pre-trained BERT layers used in Stage I. For simplicity, we refer the second option to "re-initialization".

At the t-th iteration, the teacher model generates pseudo labels $\{\widetilde{\boldsymbol{Y}}_m^{(t)} = [\widetilde{y}_{m,1}^{(t)}, ..., \widetilde{y}_{m,N}^{(t)}]\}_{m=1}^M$ by

$$\widetilde{y}_{m,n}^{(t)} = \underset{c}{\operatorname{argmax}}\, f_{n,c}(\boldsymbol{X}_m;\theta_{\text{tea}}^{(t)}). \tag{3.4}$$

Then the student model fits these pseudo-labels. Specifically, given the teacher model $f(\cdot;\theta_{\text{tea}}^{(t)})$, the student model is learned by solving

$$\widehat{\theta}_{\text{stu}}^{(t)} = \underset{\theta}{\operatorname{argmin}} \frac{1}{M}\sum_{m=1}^{M} \ell(\widetilde{\boldsymbol{Y}}_m^{(t)}, f(\boldsymbol{X}_m;\theta)). \tag{3.5}$$

We then use ADAM to optimize Eq. (3.5) with early stopping. At the end of t-th iteration, we update the teacher model and the student model by:

$$\theta_{\text{tea}}^{(t+1)} = \theta_{\text{stu}}^{(t+1)} = \widehat{\theta}_{\text{stu}}^{(t)}.$$

The algorithm is summarized in Algorithm 3.

Algorithm 3 Stage II: Self-Training

Input: M training sentences, $\{\boldsymbol{X}_m\}_{m=1}^M$; The early stopped model obtained in Stage I, $f(\cdot;\widehat{\theta})$; The number of self-training iterations T_2; The early stopping time T_3; The updating formula of ADAM $\mathcal{T}$.

1: Initialize the teacher model and the student model:

$$\theta_{\text{tea}}^{(0)} = \theta_{\text{stu}}^{(0)} = \widehat{\theta}.$$

2: **for** $t = 1, 2, ...T_2$ **do**

3: $\quad \theta_{\text{stu}}^{(t,0)} = \theta_{\text{stu}}^{(t)}$.

4: $\quad$ **for** $k = 1, 2, ..., T_3$ **do**

5: $\quad\quad$ Sample a minibatch $\mathcal{B}_k$ from $\{\boldsymbol{X}_m\}_{m=1}^M$.

6: $\quad\quad$ Generate pseudo-labels $\{\widetilde{\boldsymbol{Y}}_m\}_{m\in\mathcal{B}_k}$ by Eq. (3.4).

7: $\quad\quad$ Update the student model:

$$\theta_{\text{stu}}^{(t,k)} = \mathcal{T}(\theta_{\text{stu}}^{(t,k-1)}, \{(\boldsymbol{X}_m, \widetilde{\boldsymbol{Y}}_m)\}_{m\in\mathcal{B}_k}).$$

8: $\quad$ **end for**

9: $\quad$ Update the teacher and student:

$$\theta_{\text{tea}}^{(t)} = \theta_{\text{stu}}^{(t)} = \theta_{\text{stu}}^{(t,T_3)}.$$

10: **end for**

Output: The final student model: $\theta^{(T_2)}$

Remark 2. Note that we discard all pseudo-labels from the $(t\text{-}1)$-th iteration, and only train the student model using pseudo-labels generated by the teacher model at the t-th iteration. Combined with early stopping, such a self-training approach can improve the model fitting and reduce the noise of the pseudo-labels as illustrated in Figure 3.5. With progressive refinement of the pseudo-labels, the student model can gradually exploit knowledge in the pseudo-labels and avoid overfitting.

Remark 3. Our teacher-student framework is quite general, and can be naturally combined with other training techniques, e.g., mean teacher [65] and virtual adversarial training [59]. Please refer to Section 3.6 for more detailed discussions.

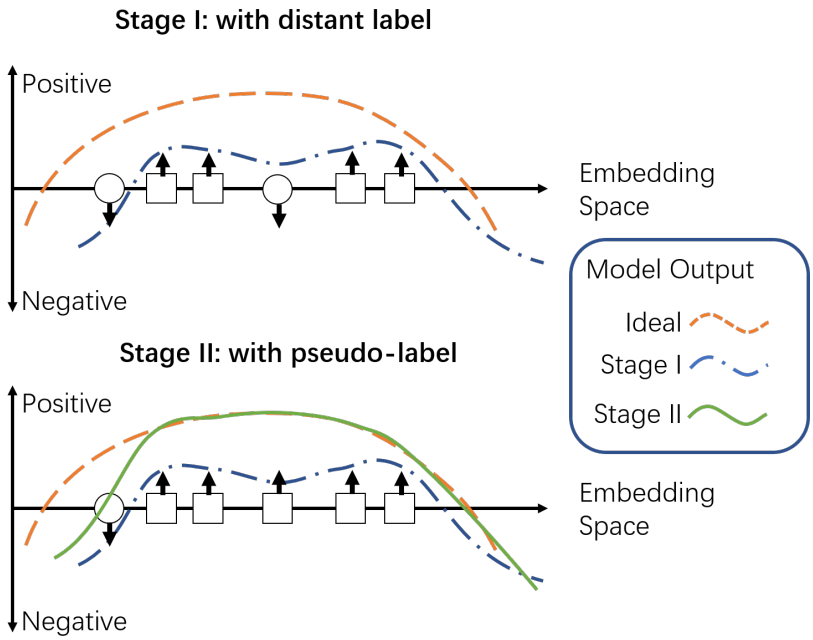


Figure 3.5: Illustration of self-training. The self-training can gradually reduce the noise of the pseudo-labels and improve model fitting.

Re-weighted High-Confidence Soft Labels

The hard pseudo-labels generated by Eq. (3.4) only keeps the most confident class for each token. To avoid losing too much information of other classes, we propose to use soft labels with confidence re-weighting.

Recall that for the n-th token in the m-th sentence, the output probability simplex over C classes is denoted as

$$[f_{n,1}(\boldsymbol{X}_m;\theta),...,f_{n,C}(\boldsymbol{X}_m;\theta)].$$

At the t-th iteration, the teacher model generates soft pseudo-labels $\{\boldsymbol{S}_m^{(t)}=[\boldsymbol{s}_{m,n}^{(t)}]_{n=1}^N\}_{m=1}^M$ following [109]:

$$\boldsymbol{s}_{m,n}^{(t)}=[s_{m,n,c}^{(t)}]_{c=1}^C=\left[\frac{f_{n,c}^2(\boldsymbol{X}_m;\theta_{\text{tea}}^{(t)})/p_c}{\sum_{c'=1}^C f_{n,c'}^2(\boldsymbol{X}_m;\theta_{\text{tea}}^{(t)})/p_{c'}}\right]_{c=1}^C \tag{3.6}$$

where $p_c=\sum_{m=1}^M\sum_{n=1}^N f_{n,c}(\boldsymbol{X}_m;\theta_{\text{tea}}^{(t)})$ calculates the unnormalized frequency of the to-

kens belonging to the c-th class. As can be seen, such a squared re-weighting step in Eq. (3.6) essentially favors the classes with higher confidence. The student model $f(\cdot;\theta_{\text{stu}}^{(t)})$ is then optimized by minimizing

$$\theta_{\text{stu}}^{(t)} = \operatorname*{argmin}_{\theta} \frac{1}{M}\sum_{m=1}^{M} \ell_{\text{KL}}(\boldsymbol{S}_m^{(t)}, f(\boldsymbol{X}_m;\theta)),$$

where $\ell_{\text{KL}}(\cdot,\cdot)$ denotes the KL-divergence-based loss:

$$\ell_{\text{KL}}(\boldsymbol{S}_m^{(t)}, f(\boldsymbol{X}_m;\theta)) = \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} -s_{m,n,c}^{(t)} \log f_{n,c}(\boldsymbol{X}_m;\theta). \tag{3.7}$$

High-Confidence Selection. To further address the uncertainty in the data, we propose to select tokens based on the prediction confidence. Specifically, at the t-th iteration, we select a set of high confidence tokens from the m-th sentence by

$$H_m^{(t)} = \{n : \max_c s_{m,n,c}^{(t)} > \epsilon\}, \tag{3.8}$$

where $\epsilon \in (0,1)$ is a tuning threshold. Accordingly, the student model $f(\cdot;\theta_{\text{stu}}^{(t)})$ can be optimized by minimizing the loss only over the selected tokens:

$$\theta_{\text{stu}}^{(t)} = \operatorname*{argmin}_{\theta} \frac{1}{M|H_m^{(t)}|}\sum_{m=1}^{M}\sum_{n\in H_m^{(t)}}\sum_{c=1}^{C} -s_{m,n,c}^{(t)} \log f_{n,c}(\boldsymbol{X}_m;\theta).$$

The high confidence selection essentially enforces the student model to better fit tokens with high confidence, and therefore is able to improve the model robustness against low-confidence tokens.

3.4 Experiments of BOND

We conduct a series of experiments to demonstrate the superiority of our proposed method.

3.4.1 Experimental Setup

Datasets

We consider the following NER benchmark datasets: (i) **CoNLL03** [110] is a well-known open-domain NER dataset from the CoNLL 2003 Shared Task. It consists of 1393 English news articles and is annotated with four entity types: person, location, organization, and miscellaneous. (ii) **Twitter** [111] is from the WNUT 2016 NER shared task. This is an open-domain NER dataset that consists of 2400 tweets (comprising 34k tokens) with 10 entity types. (iii) **OntoNotes5.0** [112] contains text documents from multiple domains, including broadcast conversation, P2.5 data and Web data. It consists of around 1.6 millions words and is annotated with 18 entity types. (iv) **Wikigold** [113] is a set of Wikipedia articles (40k tokens) randomly selected from a 2008 English dump and manually annotated with the four CoNLL03 entity types. (v) **Webpage** [114] is an NER dataset that contains personal, academic, and computer science conference webpages. It consists of 20 webpages that cover 783 entities belonging to the four types the same as CoNLL03.

For weak labels generation, we match entity types in external KBs including Wikidata corpus and gazetteers collected from multiple online sources. The data sources and matching details are described in the Appendix B.

Baselines

We compare our model with different groups of baseline methods.

• **KB Matching.** The first baseline performs string matching with external KBs using the mechanism described in the Appendix B.

• **Fully-supervised Methods.** We also include fully-supervised NER methods for comparison, including: (i) **RoBERTa-base** [16]—it adopts RoBERTa model with linear layers to perform token-level prediction; (ii) **BiLSTM-CRF** [115] adopts bi-directional LSTM with character-level CNN to produce token embeddings, which are fed into a CRF layer to

predict token labels.

• **Weakly-supervised Methods.** The third group of baselines are recent deep learning models for weakly-supervised NER, including: (i) **BiLSTM-CRF** [115] is trained using the weak labels matched from KBs; (ii) **AutoNER** [95] trains the model by assigning ambiguous tokens with all possible labels and then maximizing the overall likelihood using a fuzzy LSTM-CRF model; (iii) **LRNT** [97] is the state-of-the-art model for low-resource named tagging, which applies partial-CRFs on high-quality data with non-entity sampling. When comparing with these weakly supervised methods, we use the same weak labels as the training data for fair comparison.

• **Baselines with Different Settings**. The following methods also conduct open-domain NER under weak supervision. We remark that they use different KBs and extra training data. Therefore, we only compare with the results reported in their papers. (i) KALM [116] augments a traditional language model with a KB and use entity type information to enhance the model. (ii) ConNET [117] leverages multiple crowd annotation and dynamically aggregates them by attention mechanism. It learn from imperfect annotations from multiple sources.[4]

• For **Ablation Study**, we consider the following methods/tricks. (i) **MT** [65] uses Mean Teacher method to average model weights and forms a target-generating teacher model. (ii) **VAT** [59] adopts virtual adversarial training to smooth the output distribution to make the model robust to noise. (iii) **Hard Label** generates pseudo-labels using Eq. (3.4). (iv) **Soft Label** generates pseudo-labels using Eq. (3.6). (v) **Reinitialization** initializes the student and teacher models using Eq. (3.3). (vi) **High-Confidence Selection** selects tokens using Eq. (3.8).

[4]For KALM and ConNET model, the KB and crowd annotation are not public available, and thus we are unable to reproduce the results.

3.4.2 Experimental Results

Our NER model use RoBERTa-base as the backbone. A linear classification layer is build up on the RoBERTa-base model. Please refer to the Appendix B for implementation details.

Main Results

Table 3.2 presents the F_1 scores, precision and recall for all methods. Note that our implementations of the fully supervised NER methods attain very close to the state-of-the-art performance [8, 118]. Our results are summarized as follows:

• For all five datasets, our method consistently achieves the best performance under the weak supervision scenarios, in F_1 score, precision and recall. In particular, our method outperforms the strongest weakly supervised NER baselines by $\{11.74, 21.91, 0.66, 14.35, 12.53\}$ in terms of F_1 score. These results demonstrate the significant superiority of our proposed method.

• The standard adaptation of pre-trained language models have already demonstrated remarkable performance. The models obtained by the Stage I of our methods outperform the strongest weakly supervised NER baselines by $\{5.87, 20.51, 0.42, 7.72, 4.01\}$ in terms of F_1 score. The Stage II of our methods further improves the performance of the Stage I by $\{5.87, 1.4, 0.24, 6.63, 8.52\}$.

• On CoNLL03 dataset, compared with baselines which use different sources – KALM and ConNET, our model also outperforms them by significant margins. More detailed technical comparisons between our method and them are provided in Section 5.

Table 3.2: Main Results on Testing Set: F_1 Score (Precision/Recall) (in %)

Method	CoNLL03	Tweet	OntoNote5.0	Webpage	Wikigold
Entity Types	4	10	18	4	4
KB Matching	71.40(81.13/63.75)	35.83(40.34/32.22)	59.51(63.86/55.71)	52.45(62.59/45.14)	47.76(47.90/47.63)
Fully-Supervised (Our implementation)					
RoBERTa	90.11(89.14/91.10)	52.19(51.76/52.63)	86.20(84.59/87.88)	72.39(66.29/79.73)	86.43(85.33/87.56)
BiLSTM-CRF	91.21(91.35/91.06)	52.18(60.01/46.16)	86.17(85.99/86.36)	52.34(50.07/54.76)	54.90(55.40/54.30)
Baseline (Our implementation)					
BiLSTM-CRF	59.50(75.50/49.10)	21.77(46.91/14.18)	66.41(68.44/64.50)	43.34(58.05/34.59)	42.92(47.55/39.11)
AutoNER	67.00(75.21/60.40)	26.10(43.26/18.69)	67.18(64.63/69.95)	51.39(48.82/54.23)	47.54(43.54/52.35)
LRNT	69.74(79.91/61.87)	23.84(46.94/15.98)	67.69(67.36/68.02)	47.74(46.70/48.83)	46.21(45.60/46.84)
Other Baseline (Reported Results)					
KALM †	76.00(– / –)	–	–	–	–
ConNET◇	75.57(84.11/68.61)	–	–	–	–
Our BOND Framework					
Stage I	75.61(83.76/68.90)	46.61(53.11/41.52)	68.11(66.71/69.56)	59.11(60.14/58.11)	51.55(49.17/54.50)
BOND	81.48(82.05/80.92)	48.01(53.16/43.76)	68.35(67.14/69.61)	65.74(67.37/64.19)	60.07(53.44/68.58)

Note: †: KALM achieves better performance when using extra data. ◇: ConNET studies NER under a crowd sourcing setting, where the best human annotator achieves F_1 score at 89.51.

Ablation Study

To gain insights of our two-stage framework, we investigate the effectiveness of several components of our method via ablation study. The table 3.3 shows the results on both CoNLL03 and Wikigold datasets. Our results can be summarized as follows:

• For Stage I, **Pre-trained Language Models** significantly improve both precision and recall for both datasets. Specifically, when training the NER model from scratch, the F1 scores of the output model of Stage I drop from 75.61 to 36.66 on CoNLL03, and from 51.55 to 18.31 on Wikigold. This verifies that the rich semantic and contextual information in pre-trained RoBERTa has been successfully transferred to our NER model in Stage I.

• For Stage I, **Early stopping** improves both precision and recall for both datasets. We increase the training iterations from 900 to 18000 on CoNLL03 and from 350 to 7000 on Wikigold, and the F1 scores of the output model of Stage I drop from 75.61 to 72.11 on CoNLL03, and from 51.55 to 49.68 on Wikigold. This verifies that Early Stopping eases the overfitting and improves the generalization ability of our NER model.

• For Stage II, **Soft labels** improve the F_1 score and recall on both datasets. Specifically, the F_1 scores and recall increase from $77.28/71.98$ to $80.18/78.84$ on CoNLL03, and from $56.90/59.74$ to $58.64/65.79$ on Wikigold. Moreover, the precision on Wikigold is also improved. This verifies that the soft labels preserve more information and yield better fitted models than those of the hard labels.

• For stage II, **High-Confidence Selection** improves the F_1 scores on both datasets. Specifically, compared with using soft labels, the F_1 scores and recall increase from $81.56/78.84$ to $80.18/72.31$ on CoNLL03, and from $58.64/59.74$ to $60.07/68.58$ on Wikigold. Besides, the precision on CoNLL03 is also improved. This verifies that the high-confidence labels help select data and yield more robust performance.

• For Stage II, **Re-initialization** improves both precision and recall, only when the hard labels are adopted. We believe that this is because the hard labels lose too much information about data uncertainty, re-initializing the RoBERTa layers restores semantic and contextual

information, and can compensate such loss.

In contrast, when soft labels are adopted, **Re-initialization** deteriorates both precision and recall. We believe that this is because the soft label retains sufficient information (i.e., the knowledge transferred from RoBERTa and learned from the weak labels). As a result, re-initialization only leads to underfitting on the data.

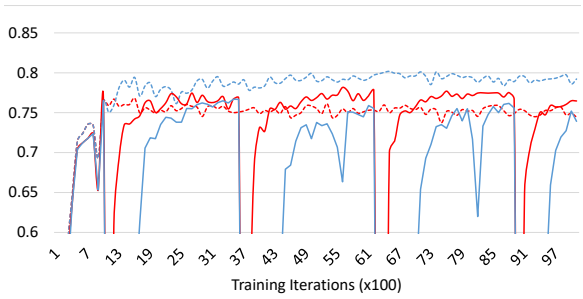
Table 3.3: Ablation Study: F_1 Score (Precision/Recall) (in %)

Method	CoNLL03	Wikigold
Stage I		
Stage I	$75.61(83.76/68.90)$	$51.55(49.17/54.50)$
Stage I w/o pre-train	$36.66(37.49/35.75)$	$18.31(18.14/18.50)$
Stage I w/o early stop	$72.11(81.65/64.57)$	$49.68(48.67/50.74)$
Stage I w/ MT	$76.30(82.92/70.67)$	$46.68(49.82/43.91)$
Stage I w/ VAT	$76.38(82.58/71.04)$	$47.54(50.02/45.30)$
Stage I + Stage II		
BOND†	$77.28(83.42/71.98)$	$56.90(54.32/59.74)$
BOND w/ soft	$80.18(81.56/78.84)$	$58.64(58.29/65.79)$
BOND w/ soft+high conf	$81.48(82.05/80.92)$	$60.07(53.44/68.58)$
BOND w/ reinit	$78.17(85.05/72.31)$	$58.55(55.31/62.19)$
BOND w/ soft+reinit	$76.92(83.39/71.38)$	$54.09(50.72/57.94)$
BOND w/ MT	$77.16(82.79/72.25)$	$57.93(55.66/60.39)$
BOND w/ VAT	$77.64(85.62/70.69)$	$57.39(55.05/59.41)$

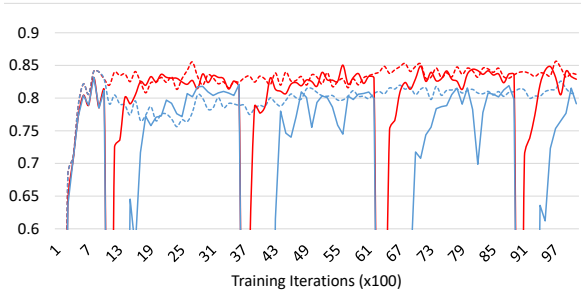
Note†*:* We use $BOND$ to denote our two-stage framework using hard pseudo-labels in this table for clarity.

Moreover, we also consider **Multiple Re-initialization**, and observe similar results.

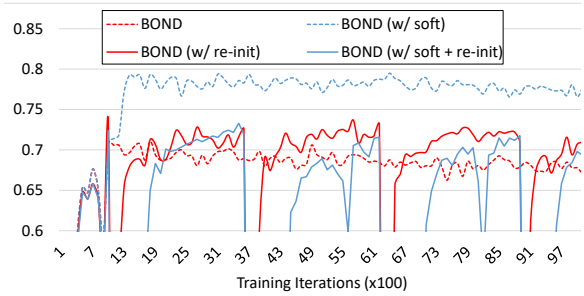
• **Mean Teacher** and **Virtual Adversarial Training** can be naturally integrated into our versatile teacher-student framework by adding an additional MT teacher or a VAT teacher. **VAT** marginally improves the F1 scores on both datasets. **MT** marginally improves the F1 scores on Wikigold, and deteriorates the F_1 scores on CoNLL03. We believe that this is because **MT** and **VAT** perform well with high quality labels, however, the labels in our NER tasks are not very precise.



(a) F_1 score



(b) Precision



(c) Recall

Figure 3.6: Learning Curves of BOND, BOND (w/ reinit), BOND (w/ soft) and BOND (w/ soft + reinit)

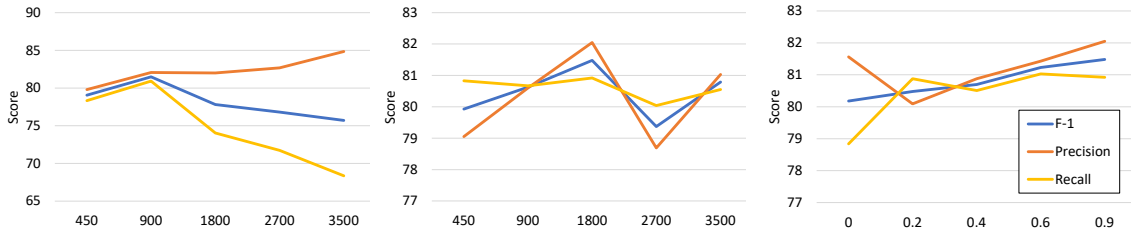
Parameter Study

We investigate the effects of the early stopping time of Stage I – T_1, the early stopping time of Stage II– T_3, and confidence threshold ϵ for selecting tokens using CoNLL03 data. The default values are $T_1 = 900, T_3 = 1800, \epsilon = 0.9$. The learning curves are summarized in Figure 3.6:

• Both T_1 and T_3 reflect trade-offs between precision and recall of the Stage I and Stage II, respectively. This verifies the importance of early stopping. The model performance is sensitive to T_1, and less sensitive to T_3.

• The recall increases along with ϵ. The precision shows a different behavior: it first decreases and then increases.

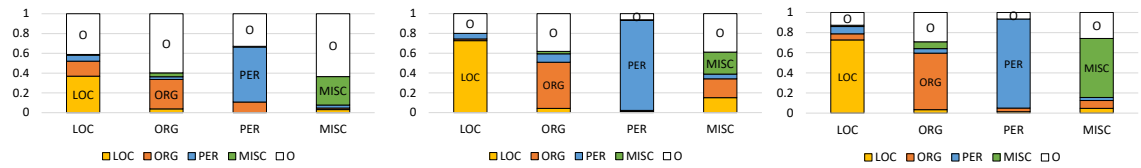
• We also consider a scenario, where T_3 is allowed to tune for each iteration of the Stage II. This requires more computational resource than the setting where T_3 remains the same for all iterations. This can further improve the model performance to 83.49, 84.09, 82.89 in terms of F_1 scores, precision and recall, respectively.



(a) The Early Stopping Time of Stage I – T_1 (b) The Early Stopping Time in Stage II – T_3 (c) The Confidence Threshold of Stage II – ϵ

Figure 3.7: Parameter Study using CoNLL03: F_1, Precision, Recall on Testing Set (in %)

Case Study and Error Analysis



(a) Knowledge Base Matching (b) Stage I (c) Stage II

Figure 3.8: Recall of Knowledge Base Matching and different stages of BOND. The horizontal axis denotes the true entity type. The segments in a bar denote the portions of the entities being classified into different entity types.

To demonstrate how BOND improves the recall, we compare the prediction performance of KB matching with the output models of Stage I and Stage II using Wikigold data. Figure 3.8 presents the bar plots of four entity types – "`LOC`", "`PER`", "`ORG`" and "`MISC`". As can be seen, the KB matching yields a large amount of "`O`" (non-entity) due to its limited coverage. As a result, the recall is very low 47.63%. In contrast, our model of the

Stage I benefits from the transferred knowledge of pre-trained RoBERTa and is able to correct some wrongly matched O's to their corresponding entity types. Therefore, it enjoys a better recall 54.50%. Moreover, the self-training in the Stage II further improves the recall to 68.48%.

3.5 Extension: Named Entity Recognition with Small Strongly Labeled and Large Weakly Labeled Data

In practice, we often can access both a small amount of strongly labeled data and a large amount of weakly labeled data, generated from large scale unlabeled data and domain knowledge bases. A natural question arises here:

"Can we simultaneously leverage small strongly and large weakly labeled data to improve the model performance?"

The answer is yes, but the prerequisite is that you can properly suppress the extensive labeling noise in the weak labels.

An ultra-large volume of weakly labeled data contains useful domain knowledge. But it also comes with enormous noise due to the "incompleteness" and "labeling bias" of weak labels. The enormous noise can dominate the signal in the strongly and weakly labeled data, especially when combined with the unsupervised pre-training techniques. Such noise can be easily overfitted by the huge neural language models, and may even deteriorate the model performance. This is further corroborated by our empirical observation that when we train deep NER models over a simple or weighted combination of the strongly labeled and weakly labeled data, the model performance almost always becomes worse.

To leverage the ultra-large weakly labeled data, we propose a variant of BOND, which is a three-stage computational framework named NEEDLE (**N**oise-aware w**E**akly sup**E**rvise**D** continua**L** pr**E**-training).

Our experimental results show that NEEDLE significantly improves the model performance on the E-commerce query NER tasks and Biomedical NER tasks. In particular,

we achieve new SOTA F1-scores on 3 Biomedical NER datasets: BC5CDR-chem 93.74, BC5CDR-disease 90.69, NCBI-disease 92.28. We also extend the proposed framework to the multi-lingual setting.

3.5.1 Three-stage Computational Framework – NEEDLE

To harness the power of weakly labeled data, we propose a new framework — NEEDLE, which contain stages as illustrated in Figure 3.9:

1) We first adapt an open-domain pre-trained language model to the downstream domain via MLM continual pre-training on the unlabeled in-domain data.

2) We use the knowledge bases to convert the unlabeled data to the weakly labeled data through weak supervision. Then we apply noise-aware continual pre-training for learning task-specific knowledge from both strongly and weakly labeled data;

3) Lastly, we fine-tune the model on the strongly labeled data again.

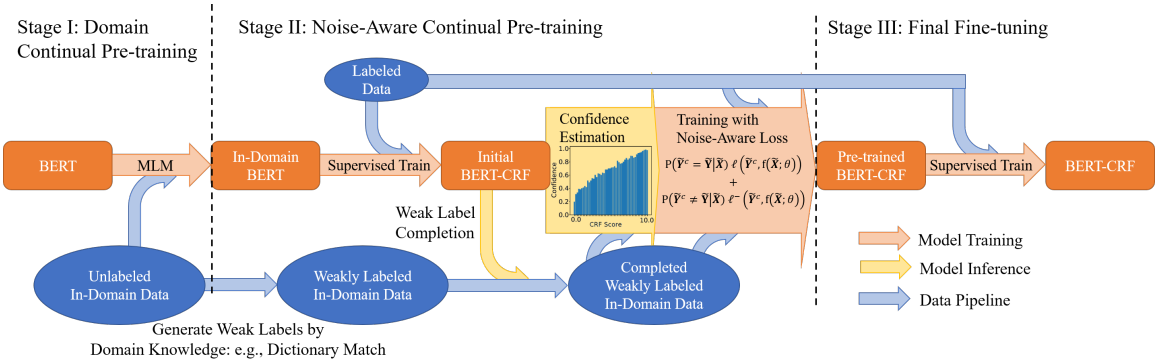


Figure 3.9: Three-stage NEEDLE Framework.

Stage I: Domain Continual Pre-training over Unlabeled Data

Following previous work on domain-specific BERT [119, 120], we first conduct domain continual masked language model pre-training on the large in-domain unlabeled data $\{\widetilde{\boldsymbol{X}}_m\}_{m=1}^{\widetilde{M}}$. Note that the masked language model $f_{\text{LM}}(\cdot; \theta_{\text{enc}}, \theta_{\text{LM}})$ contains encoder parameters θ_{enc} and classification head parameters θ_{LM}, which are initialized from open-domain pre-trained masked language models (e.g., BERT and RoBERTa).

Stage II: Noise-Aware Continual Pre-training over both Strongly and Weakly labeled Data

In the second stage, we use the knowledge bases to convert the unlabeled data to weakly labeled data to generate weak labels for the unlabeled data: $\{(\widetilde{\boldsymbol{X}}_m, \widetilde{\boldsymbol{Y}}_m^w)\}_{m=1}^{\widetilde{M}}$. We then continually pre-train the model with both weakly labeled in-domain data and strongly labeled data. Specifically, we first replace the MLM head by a CRF classification head [121] and conduct noise-aware weakly supervised learning, which contains two ingredients: *weak label completion procedure* and *noise-aware loss function*.

• **Weak Label Completion**. As the weakly labeled data suffer from severe missing entity issue, we propose a weak label completion procedure. Specifically, we first train an initial NER model $f(;\theta^{\text{Init}})$ by minimizing the following loss over $\{(\boldsymbol{X}_m, \boldsymbol{Y}_m)\}_{m=1}^{M}$:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{\mathrm{M}} \ell(\boldsymbol{Y}_m, f(\boldsymbol{X}_m; \theta)), \tag{3.9}$$

where $\ell(\cdot, \cdot)$ is the cross-entropy loss for token-wise classification model or negative likelihood for CRF model [121], and the model parameters is $\theta^{\text{Init}} = (\theta_{\text{enc}}, \theta_{\text{CRF}})$ with the encoder θ_{enc} initialized from Stage I and NER CRF head θ_{CRF} initialized randomly. Then, for a given sentence $\widetilde{\boldsymbol{X}} = [x_1, ..., x_N]$ with the original weak labels $\widetilde{\boldsymbol{Y}}^w = [y_1^w, ..., y_N^w]$ and the predictions from the initial model $\widetilde{\boldsymbol{Y}}^p = \operatorname{argmin}_{\boldsymbol{Y}} \ell(\boldsymbol{Y}, f(\widetilde{\boldsymbol{X}}; \theta^{\text{Init}})) = [y_1^w, ..., y_N^w]$, we generate the corrected weak labels $\widetilde{\boldsymbol{Y}}^c = [y_1^c, ..., y_N^c]$ by:

$$y_i^c = \begin{cases} y_i^p & \text{if } y_i^w = \text{O (non-entity)} \\ y_i^w & \text{otherwise} \end{cases} \tag{3.10}$$

Such a weak label completion procedure can remedy the incompleteness of weak labels.

• **Noise-Aware Loss Function**. The model tends to overfit the noise of weak labels when using negative log-likelihood loss over the weakly labeled data, Eq.(3.9). To alleviate this issue, we propose a noise-aware loss function based on the confidence of the corrected weak

labels $\widetilde{\mathbf{Y}}^c$, which is defined as the probability of $\widetilde{\mathbf{Y}}^c$ being the true labels $\widetilde{\mathbf{Y}}$: $P(\widetilde{\mathbf{Y}}^c = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}})$. The confidence can be estimated by the model prediction score $f(\widetilde{\mathbf{X}};\theta)$ and histogram binning [122]. See more details in Appendix B.4.1.

We design the noise-aware loss function to make the fitting to the weak labels more conservative/aggressive, when the confidence is lower/higher. Specifically, when $\widetilde{\boldsymbol{Y}}^c = \widetilde{\boldsymbol{Y}}$, we let loss function $\mathcal{L}$ be the negative log-likelihood, i.e., $\mathcal{L}(\cdot,\cdot|\widetilde{\boldsymbol{Y}}^c = \widetilde{\boldsymbol{Y}}) = \ell(\cdot,\cdot)$; when $\widetilde{\boldsymbol{Y}}^c \neq \widetilde{\boldsymbol{Y}}$, we let $\mathcal{L}$ be the negative log-unlikelihood, i.e., $\mathcal{L}(\cdot,\cdot|\widetilde{\boldsymbol{Y}}^c \neq \widetilde{\boldsymbol{Y}}) = \ell^-(\cdot,\cdot)$ [5]. Accordingly, the noise-aware loss function is designed as

$$\begin{aligned}
&\ell_{\mathrm{NA}}(\widetilde{\boldsymbol{Y}}^c, f(\widetilde{\boldsymbol{X}};\theta)) \\
&= \mathbb{E}_{\widetilde{\boldsymbol{Y}}_m=\widetilde{\boldsymbol{Y}}_m^c|\widetilde{\boldsymbol{X}}_m}\mathcal{L}(\widetilde{\boldsymbol{Y}}_m^c, f(\widetilde{\boldsymbol{X}}_m;\theta), \mathbb{1}(\widetilde{\boldsymbol{Y}}_m = \widetilde{\boldsymbol{Y}}_m^c)) \\
&= P(\widetilde{\boldsymbol{Y}}^c = \widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}})\ell(\widetilde{\boldsymbol{Y}}^c, f(\widetilde{\boldsymbol{X}};\theta)) + \\
&\quad P(\widetilde{\boldsymbol{Y}}^c \neq \widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}})\ell^-(\widetilde{\boldsymbol{Y}}^c, f(\widetilde{\boldsymbol{X}};\theta)),
\end{aligned} \tag{3.11}$$

where the log-unlikelihood loss can be viewed as regularization and the confidence of weak labels can be viewed as an adaptive weight. The training objective on both the strongly labeled data and weakly labeled data is:

$$\begin{aligned}
\min_{\theta} \frac{1}{M+\widetilde{M}}[&\sum_{m=1}^{M}\ell(\boldsymbol{Y}_m, f(\boldsymbol{X}_m;\theta)) \\
&+\sum_{m=1}^{\widetilde{M}}\ell_{\mathrm{NA}}(\widetilde{\boldsymbol{Y}}_m^c, f(\widetilde{\boldsymbol{X}}_m;\theta))],
\end{aligned} \tag{3.12}$$

Stage III: Final Fine-tuning

Stages I and II of our proposed framework mainly focus on preventing the model from the overfitting to the noise of weak labels. Meanwhile, they also suppress the model fitting to the strongly labeled data. To address this issue, we propose to fine-tune the model on

[5] $\ell(\boldsymbol{Y}, f(\boldsymbol{X};\theta)) = -\log P_{f(\boldsymbol{X};\theta)}(\boldsymbol{Y})$
$\ell^-(\boldsymbol{Y}, f(\boldsymbol{X};\theta)) = -\log\left[1 - P_{f(\boldsymbol{X};\theta)}(\boldsymbol{Y})\right]$

the strongly labeled data again. Our experiments show that such additional fine-tuning is essential.

3.5.2 Experiments

We use transformer-based open-domain pretrained models, e.g., BERT, mBERT, RoBERTa-Large, [8, 16] with a CRF layer as our base NER models. Throughout the experiments, we use the `BIO` tagging scheme [123]. For Stages I and II, we train the models for one epoch with batch size 144. For Stage III, we use the grid search to find optimal hyperparameters: We search the number of epochs in $[1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 50]$ and batch size in $[64, 144, 192]$. We use ADAM optimizer with a learning rate of 5×10^{-5} on the E-commerce query NER dataset. In the Biomedical NER experiments, we search the optimal learning rate in $[1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}]$. All implementations are based on *transformers* [124]. We use an Amazon EC2 virtual machine with 8 NVIDIA V100 GPUs.

Table 3.4: Data Statistics

Dataset	Number of Samples				Weak Label	
	Train	Dev	Test	Weak	Precision	Recall
E-commerce Query Domain						
En	187K	23K	23K	22M	84.62	49.52
E-commerce Query Domain (Multilingual)						
Mul-En	257K	14K	14K			
Mul-Fr	79K	4K	4K			
Mul-It	52K	3K	3K	17M	84.62	49.52
Mul-De	99K	5K	5K			
Mul-Es	64K	4K	4K			
Biomedical Domain						
BC5CDR Chem	5K	5K	5K	11M	92.08	77.40
BC5CDR Disease	5K	5K	5K	15M	94.46	81.34
NCBI Disease	5K	1K	1K			

Datasets

We evaluate the proposed framework on two different domains: E-commerce query domain and Biomedical domain. The data statistics are summarized in Table 3.4.

For E-commerce query NER, we consider two settings: english queries and multilingual queries. For English NER, there are 10 different entity types, while the multilingual NER has 12 different types. The queries are collected from search queries to a shopping website. The unlabeled in-domain data and the weak annotation is obtained by aggregating user behavior data collected from the shopping website.

For Biomedical NER, we use three popular benchmark datasets: BC5CDR-Chem, BC5CDR-Disease [125], and NCBI-Disease [126]. These datasets only contain a single entity type. We use the pre-processed data in `BIO` format from Crichton et al. [127] following BioBERT [120] and PubMedBERT [128]. We collect unlabeled data from PubMed 2019 baseline [6], and use the dictionary lookup and exact string match to generate weak labels [7]. We only include sentences with at least one weak entity label.

• **Weak Labels Performance**. Table 3.4 also presents the precision and recall of weak labels performance on a evaluation golden set. As can be seen, the weak labels suffer from severe incompleteness issue. In particular, the recall of E-commerce query NER is lower than 50. On the other hand, the weak labels also suffer from labeling bias.

Baselines

We compare NEEDLE with the following baselines (All pre-trained models used in the baseline methods have been continually pre-trained on the in-domain unlabeled data (i.e., Stage I of NEEDLE) for fair comparison):

• Supervised Learning Baseline: We directly fine-tune the pre-trained model on the strongly labeled data. For E-commerce query NER, we use Query-RoBERTa-CRF, which is adapted

[6]Titles and abstract of Biomedical articles:https://ftp.ncbi.nlm.nih.gov/pubmed/baseline/

[7]We collect a dictionary containing 3016 chemical entities and 5827 disease entities.

from the RoBERTa large model. For E-commerce multilingual query NER, we use Query-mBERT-CRF, which is adapted from the mBERT. For Biomedical NER, we use BioBERT-CRF [120], which is adapted from BERT-base.

• Semi-supervised Self-Training (SST): SST use the model obtained by supervised learning to generate pseudo labels for the unlabeled data and then conduct semi-supervised leaning [129, 130].

• Weakly Supervised Learning (WSL): Simply combining strongly labeled data with weakly labeled data [131].

• Weighted WSL: WSL with weighted loss, where weakly labeled samples have a fixed different weight. We tune the weight and present the best result.

• Robust WSL: WSL with mean squared error loss function, which is robust to label noise [132]. As the robust loss is not compatible with CRF, we use the token-wise classification model for the Stage II training.

• Partial WSL: WSL with non-entity weak labels excluded from the training loss [95].

Table 3.5: Main Results on E-commerce English Query NER: Span-level Precision/Recall/F1. †: we presented the results of the best weight, see results for all weights in Appendix B.4.2.

Method	P	R	F1
NEEDLE	**80.71**	**80.55**	**80.63**
Supervised Baseline			
Query-RoBERTa-CRF	79.27	79.24	79.25
Semi-supervised Baseline			
SST	79.61	79.37	79.75
Weakly Supervised Baselines			
WSL	73.95	50.20	59.81
Weighted WSL †	78.07	64.41	70.59
Partial WSL	71.95	68.56	70.21
Weighted Partial WSL †	76.28	76.34	76.31
Robust WSL	66.71	42.78	52.13

E-commerce NER

We use span-level precision/recall/F1-score as the evaluation metrics. We present the main results on English query NER in Table 3.5.

Main Results

• **NEEDLE**: NEEDLE outperforms the fully supervised baseline and achieves the best performance among all baseline methods;

• **Weakly Supervised Baselines**: All weakly supervised baseline methods, including WSL, Weighted WSL, Partial WSL and Robust WSL, lead to worse performance than the supervised baseline. This is consistent with our claim in Section 1. The weakly labeled data can hurt the model performance if they are not properly handled;

• **SST**: Semi-supervised self-training outperforms the supervised baseline and weakly supervised baselines. This indicates that if not properly handled, the weak labels are even worse than the pseudo label generated by model prediction. In contrast, NEEDLE outperforms SST, which indicates that the weak labels can indeed provide additional knowledge and improve the model performance when their noise can be suppressed.

Ablation

We study the effectiveness of each component of NEEDLE. Specifically, we use the following abbreviation to denote each component of NEEDLE:

• WLC: Weak label completion.

• NAL: Noise-aware loss function, i.e., Eq.(3.12). Since NAL is built on top of WLC, the two components need to be used together.

• FT: Final fine-tuning on strongly labeled data (Stage III).

As can be seen from Table 3.6, all components are effective, and they are complementary to each other.

Table 3.6: Ablation Study on E-commerce English Query NER.

Method	P	R	F1
NEEDLE w/o FT/WLC/NAL	73.95	50.20	59.81
NEEDLE w/o FT/NAL	75.53	76.45	75.99
NEEDLE w/o FT	75.86	76.56	76.21
NEEDLE w/o WLC/NAL	80.03	79.72	79.87
NEEDLE w/o NAL	80.07	80.36	80.21
NEEDLE	**80.71**	**80.55**	**80.63**

Extension to Multilingual NER

The proposed framework can be naturally extended to improve multilingual NER. See details about the algorithm in Appendix B.4.4. The results of E-commerce Multilingual NER is presented in Table 3.7. As can be seen, the proposed NEEDLE outperforms other baseline methods in all 5 languages.

Table 3.7: E-commerce Multilingual Query NER: Span Level F1. See other metrics in Appendix B.4.4.

Method	En	Fr	It	De	Es
NEEDLE	**78.17**	**75.98**	**79.68**	**78.83**	**79.49**
Supervised Baseline					
Query-mBERT-CRF	77.19	74.82	78.11	77.77	78.11
Semi-supervised Baseline					
SST	77.42	75.21	77.82	78.10	78.65
Weakly supervised Baseline					
WSL	58.35	59.90	60.98	61.66	63.14

Biomedical NER

We present the main results on Biomedical NER in Table 3.8. NEEDLE achieves the best performance among all comparison methods. We outperform previous SOTA [120, 128] by 0.41%, 5.07%, 3.15%, on BC5CDR-chemical, BC5CDR-disease and NCBI-disease respectively, in terms of the F1-score. We achieve very significant improvement on BC5CDR-disease. We conjecture that the weak labels for disease entities are relatively accurate, since

WSL can also improve the model performance.

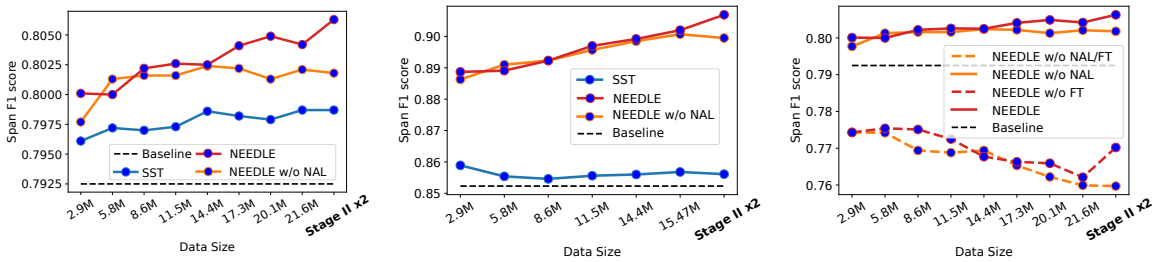
Table 3.8: Main Results on Biomedical NER: Span Level F1-score. We also provide previous SOTA performance reported in Gu et al. [128] and Nooralahzadeh, Lønning, and Øvrelid [133].. †: NER-PA-RL is a WSL variant using instance selection. Nooralahzadeh, Lønning, and Øvrelid [133] only report the averaged F1 of BC5CDR-chemical and BC5CDR-disease. See other metrics in Appendix B.4.3.

Method	BC5CDR chemical	BC5CDR disease	NCBI disease
NEEDLE	**93.74**	**90.69**	**92.28**
Supervised Baseline			
BioBERT-CRF	92.96	85.23	89.22
Semi-supervised Baseline			
SST	93.06	85.56	89.42
Weakly-supervised Baseline			
WSL	85.41	88.96	78.84
Reported F1-scores in [128].			
BERT	89.99	79.92	85.87
BioBERT	92.85	84.70	89.13
SciBERT	92.51	84.70	88.25
PubMedBERT	93.33	85.62	87.82
Reported F1-scores in [133].			
NER-PA-RL†	89.93		-

Analysis

Size of Weakly Labeled Data. To demonstrate that NEEDLE can better exploit the weakly labeled data, we test the model performance with randomly sub-sampled weakly labeled data. We plot the F1-score curve for E-commerce English query NER in Figure 3.10a and BC5CDR data in Figure 3.10b. We find that NEEDLE gains more benefits from increasing the size of weakly labeled data compared with other methods (SST and WSL). We also present the performance of NEEDLE w/o FT in Figure 3.10c. As can be seen, although the performance of NEEDLE w/o FT decreases with more weakly labeled data, the model can still learn more useful information and achieves better performance after fine-tuning.

Two Rounds of Stage II Training. Since the model after the final fine-tuning is better



(a) E-commerce En Query NER (b) BC5CDR-disease (c) w/ vs. w/o final fine-tuning

Figure 3.10: Size of weakly labeled data vs. Performance. We present the performance after the final round of fine-tuning in (a) and (b). We also compare the performance with and without fine-tuning in (c) using E-commerce English query NER data. The baselines are Query-RoBERTa-CRF for (a,c) and BioBERT-CRF for (b). "Baseline": the baseline here is the fully supervised baseline. We also present the performance after two rounds of Stage II training at the rightmost point of each curve ("*Stage II x2*").

than the initial model in Stage II, we study whether using the fine-tuned model for an addition round of Stage II can further improve the performance of NEEDLE. Specifically, after Stage III, we 1) use the new model to complete the original weak labels; 2) conduct noise-aware continual pre-training over both strongly and weakly labeled data; 3) fine-tune the model on strongly labeled data. The results are presented in Figure 3.10 (last point of each curve). As can be seen, NEEDLE can obtain slight improvement using the two rounds of Stage II training. On the other hand, we also show that SST and NEEDLE w/o NAL achieve little improvement using the second round of training.

Size of Strongly Labeled Data. To demonstrate that NEEDLE is sample efficient, we test NEEDLE on randomly sub-sampled strongly labeled data on E-commerce NER. As we show in Figure 3.11, NEEDLE only requires $30\% \sim 50\%$ strongly labeled data to achieve the same performance as the (fully) supervised baseline. We also observe that NEEDLE achieves more significant improvement with fewer labeled data: +2.28/3.64 F1-score with 1%/10% labeled data.

Weak Label Errors in E-commerce NER

Here we study several possible errors of the weak labels to better understand the weak labels and how the proposed techniques reduce these errors.

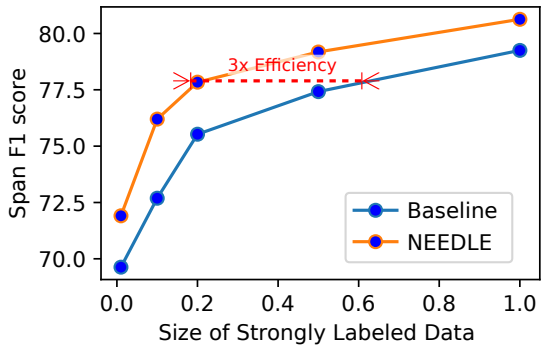


Figure 3.11: Performance vs. Size of Strongly Labeled Data. See detailed numbers in Appendix B.4.2.

Table 3.9: Query Examples of "amiibo". Entity Labels: Red: Misc, Blue: Product Line, Green: Color, Black: Non Entity, Orange: Media Title.

Label Types	Querys and Labels			
Human Labels	zelda amiibo	wario amiibo	yarn yoshi amiibo	amiibo donkey kong
Original Weak Labels	zelda amiibo	wario amiibo	yarn yoshi amiibo	amiibo donkey kong
Corrected Weak Labels	zelda amiibo	wario amiibo	yarn yoshi amiibo	amiibo donkey kong
Self-Training Labels	zelda amiibo	wario amiibo	yarn yoshi amiibo	amiibo donkey kong

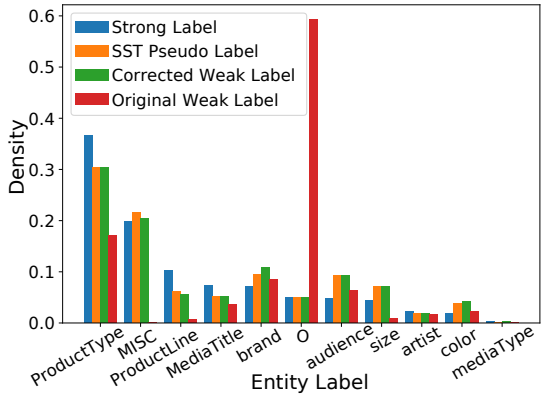


Figure 3.12: Entity Distribution

Label Distribution Mismatch. First, we show the distribution difference between the weak labels and the strong labels, and demonstrate how the weak label completion reduces the gap. Specifically, we compare the entity distribution of the true labels, weak labels, corrected weak labels and self-training pseudo labels in Figure 3.12. As can be seen, the original weak labels suffer from severe missing entity issue (i.e., too many non-entity labels) and distribution shift (e.g., nearly no `Misc` labels). On the other hand, the corrected

weak labels suffer less from the missing entities and distribution shift. SST pseudo labels are the most similar to the strong labels, which explains why SST can directly improves the performance.

Systematical Errors. We observe that many errors from the weakly labeled data are systematical errors, which can be easily fixed by the final fine-tuning stage. For example, "amiibo" is one `Product Line` of "nintendo". The amiibo characters should be defined as `Misc` type, while the weak labels are all wrongly annotated as `Color`. We list 4 queries and their strong labels and weak labels in Table 3.9. Although these errors lead to worse performance in Stage II, they can be easily fixed in the final fine-tuning stage. Specifically, the pre-training first encourages the model to learn that "xxx amiibo" is a combination of `color + productLine` with a large amount of weakly labeled data, and then the fine-tuning step corrects such a pattern to `misc + productLine` with a limited amount of data. It is easier than directly learning the `misc + productLine` with the limited strongly labeled data.

Entity `BIO` Sequence Mismatch in Weak Label Completion. Another error of the weakly labels is the mismatched entity `BIO` sequence in the weak label competion step, e.g., `B-productType` followed by `I-color` [8]. For English Query NER, the proportion of these broken queries is 1.39%. Removing these samples makes the Stage II perform better (F1 score +1.07), while it does not improve the final stage performance (F1 score -0.18e). This experiment indicates that the final fine-tuning suffices to correct these errors, and we do not need to strongly exclude these samples from Stage II.

Quantify the Impact of Weak Labels. Here we examine the impact of weak labels via the lens of prediction error. We check the errors made by the model on the validation set. There are 2384 entities are wrongly classified by the initial NER model. After conducting NEEDLE, 454 of 2384 entities are correctly classified. On the other hand, the model makes 311 more wrong predictions. Notice that not all of them are directly affected by the weakly

[8] E.g., Original Weak Labels: `B-productType, O, O`; Model Prediction: `B-color,I-color,O`; Corrected Weak Labels: `B-productType, I-color, O`.

labeled data, i.e., some entities are not observed in the weakly labeled data. Some changes may be only due to the data randomness. If we exclude the entities which are not observed in the weakly annotated entities, there are 171 new correctly classified entities and 93 new wrongly classified entities, which are affected by the weak labels. Such a ratio $171/93 = 1.84 >> 1$ justifies that the advantage of NAL significantly out-weights the disadvantage of the noise of weak labels.

3.6 Related Work and Discussion

Our work is related to **low-resource NER**. This line of research focuses on leveraging cross lingual information to improve the model performance. For examples, [134, 135] consider NER for a low resource target language. They propose to train an NER model with annotated language that are closely related to the target language. [136] propose to use the bilingual dictionaries to tackle this challenge. More recently, [137] propose a Bayesian graphical model approach to further improve the low resource NER performance.

Our work is also relevant to **semi-supervised learning**, where the training data is only partially labeled. There have been many semi-supervised learning methods, including the popular Mean Teacher and Virtual Adversarial Training methods used in our experiments for comparison [35, 65, 59, 138, 139]. Different from weak supervision, these semi-supervised learning methods usually has a partial set of labeled data. They rely on the labeled data to train an sufficiently accurate model. The unlabeled data are usually used for inducing certain regularization to further improve the generalization performance. The weak supervision, however, considers the setting with only noisy labels. Existing semi-supervised learning methods such as Mean Teacher and Virtual Adversarial Training can only marginally improve the performance, as shown in the ablation study in our experiments.

Other related works: [116] propose a language model-based method — KALM for NER tasks. However, their approach has two drawbacks: (i) Since they design a language model designated for NER tasks, they need to first train the language models from scratch. How-

ever, this often requires a large amount of training corpus and enormous computational resources. In contrast, BOND uses general-purpose pre-trained language models, which are publicly available online. (ii) The training of their language model is not fully unsupervised and requires token-level annotations. To address this issue, they resort to weak supervision, which yields incomplete and noisy annotations. Therefore, their language model does not necessarily achieve the desired performance.

Larger Pre-trained Language Models: To further improve the performance of BOND, we can use larger pre-trained language models such as RoBERTa-large [16] (Three times as big as RoBERT-base in our experiments) and T5 [99] (Thirty times larger than RoBERTa-base). These larger models contain more general semantics and syntax information, and have the potentials to achieve even better performance for NER Tasks. Unfortunately, due to the limitation of our computational resources, we are unable to use them in our experiments.

CHAPTER 4
WEAKLY SUPERVISED LEARNING WITH TRANSFER LEARNING AND CONTRASTIVE-REGULARIZED SELF-TRAINING

This chapter extends Chapter 3 to more NLP tasks and further improves the performance via contrastive-regularized self-training. The content is based on the following publication:

Yue Yu et al. (2021). "Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*

4.1 Overview

We propose a new framework COSINE[1] that fine-tunes pre-trained LMs with only weak supervision. COSINE leverages both weakly labeled and unlabeled data, as well as suppresses label noise via contrastive self-training. Weakly-supervised learning enriches data with potentially noisy labels, and our contrastive self-training scheme fulfills the denoising purpose. Specifically, contrastive self-training regularizes the feature space by pushing samples with the same pseudo-labels close while pulling samples with different pseudo-labels apart. Such regularization enforces representations of samples from different classes to be more distinguishable, such that the classifier can make better decisions. To suppress label noise propagation during contrastive self-training, we propose confidence-based sample reweighting and regularization methods. The reweighting strategy emphasizes samples with high prediction confidence, which are more likely to be correctly classified, in order to reduce the effect of wrong predictions. Confidence regularization encourages smoothness over model predictions, such that no prediction can be over-confident, and therefore

[1] Short for **Co**ntrastive **S**elf-Training for F**ine**-Tuning Pre-trained Language Model.

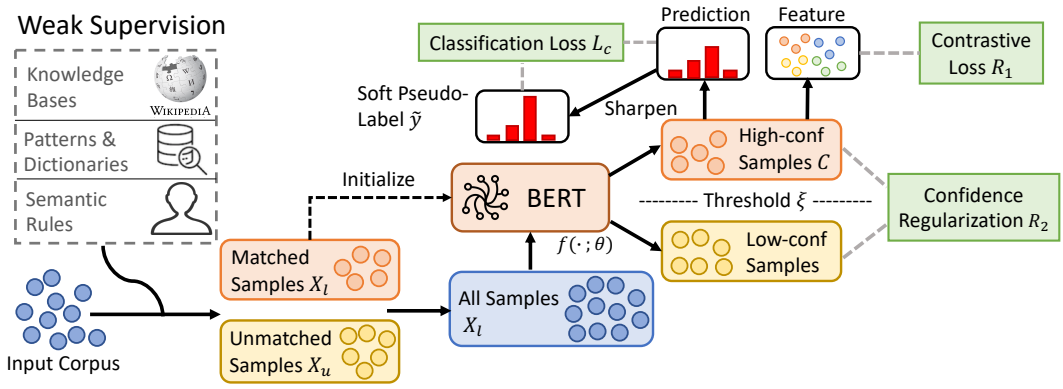


Figure 4.1: The framework of COSINE. We first fine-tune the pre-trained language model on weakly-labeled data with early stopping. Then, we conduct contrastive-regularized self-training to improve model generalization and reduce the label noise. During self-training, we calculate the confidence of the prediction and update the model with high confidence samples to reduce error propagation.

reduces the influence of wrong pseudo-labels.

Our model is flexible and can be naturally extended to semi-supervised learning, where a small set of clean labels is available. Moreover, since we do not make assumptions about the nature of the weak labels, COSINE can handle various types of label noise, including biased labels and randomly corrupted labels. Biased labels are usually generated by semantic rules, whereas corrupted labels are often produced by crowd-sourcing.

Extensive experiments on 6 NLP classification tasks using 7 public benchmarks verifying the efficacy of COSINE. We highlight that our model achieves competitive performance in comparison with fully-supervised models on some datasets, e.g., on the Yelp dataset, we obtain a 97.2% (fully-supervised) v.s. 96.0% (ours) accuracy comparison.

4.2 Preliminary

In this section, we introduce weak supervision and our problem formulation.

Weak Supervision. Instead of using human-annotated data, we obtain labels from weak supervision sources, including keywords and semantic rules[2]. From weak supervision sources, each of the input samples $x \in \mathcal{X}$ is given a label $y \in \mathcal{Y} \cup \{\emptyset\}$, where $\mathcal{Y}$ is

[2]Examples of weak supervisions are in Appendix C.1.3.

the label set and $\emptyset$ denotes the sample is not matched by any rules. For samples that are given multiple labels, e.g., matched by multiple rules, we determine their labels by majority voting.

Problem Formulation. We focus on the weakly-supervised classification problems in natural language processing. We consider three types of tasks: sequence classification, token classification, and sentence pair classification. These tasks have a broad scope of applications in NLP.

Formally, the weakly-supervised classification problem is defined as the following: Given weakly-labeled samples $\mathcal{X}_l = \{(x_i, y_i)\}_{i=1}^{L}$ and unlabeled samples $\mathcal{X}_u = \{x_j\}_{j=1}^{U}$, we seek to learn a classifier $f(x;\theta) : \mathcal{X} \to \mathcal{Y}$. Here $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$ denotes all the samples and $\mathcal{Y} = \{1, 2, \cdots, C\}$ is the label set, where C is the number of classes.

4.3 Method

Our classifier $f = g \circ \text{BERT}$ consists of two parts: BERT is a pre-trained language model that outputs hidden representations of input samples, and g is a task-specific classification head that outputs a C-dimensional vector, where each dimension corresponds to the prediction confidence of a specific class. In this chapter, we use RoBERTa [16] as the realization of BERT.

The framework of COSINE is shown in Figure 4.1. First, COSINE initializes the LM with weak labels. In this step, the semantic and syntactic knowledge of the pre-trained LM are transferred to our model. Then, it uses contrastive self-training to suppress label noise propagation and continue training.

4.3.1 Overview

The training procedure of COSINE is as follows.

Initialization with Weakly-labeled Data. We fine-tune $f(\cdot;\theta)$ with weakly-labeled data

$\mathcal{X}_l$ by solving the optimization problem

$$\min_{\theta} \frac{1}{|\mathcal{X}_l|} \sum_{(x_i,y_i)\in\mathcal{X}_l} \text{CE}\left(f(x_i;\theta), y_i\right), \tag{4.1}$$

where $\text{CE}(\cdot,\cdot)$ is the cross entropy loss. We adopt early stopping [141] to prevent the model from overfitting to the label noise. However, early stopping causes underfitting, and we resolve this issue by contrastive self-training.

Contrastive Self-training with All Data. The goal of contrastive self-training is to leverage all data, both labeled and unlabeled, for fine-tuning, as well as to reduce the error propagation of wrongly labelled data. We generate pseudo-labels for the unlabeled data and incorporate them into the training set. To reduce error propagation, we introduce contrastive representation learning (Sec. 4.3.2) and confidence-based sample reweighting and regularization (Sec. 4.3.3). We update the pseudo-labels (denoted by $\widetilde{\boldsymbol{y}}$) and the model iteratively. The procedures are summarized in Algorithm 4.

$\diamond$ **Update $\widetilde{\boldsymbol{y}}$ with the current θ.** To generate the pseudo-label for each sample $x \in \mathcal{X}$, one straight-forward way is to use hard labels [36]

$$\widetilde{y}_{\text{hard}} = \underset{j\in\mathcal{Y}}{\text{argmax}} \left[f(x;\theta)\right]_j. \tag{4.2}$$

Notice that $f(x;\theta) \in \mathbb{R}^C$ is a probability vector and $[f(x;\theta)]_j$ indicates the j-th entry of it. However, these hard pseudo-labels only keep the most likely class for each sample and result in the propagation of labeling mistakes. For example, if a sample is mistakenly classified to a wrong class, assigning a 0/1 label complicates model updating (Eq. 4.4), in that the model is fitted on erroneous labels. To alleviate this issue, for each sample x in a batch $\mathcal{B}$, we generate soft pseudo-labels[3] [142, 23, 143] $\widetilde{\boldsymbol{y}} \in \mathbb{R}^C$ based on the current

[3]More discussions on hard vs.soft are in Sec. 4.4.5.

Algorithm 4 Training Procedures of COSINE.

Input: Training samples $\mathcal{X}$; Weakly labeled samples $\mathcal{X}_l \subseteq \mathcal{X}$; Pre-trained LM $f(\cdot;\theta)$.

// Fine-tune the LM with weakly-labeled data.

for $t = 1, 2, \cdots, T_1$ **do** Sample a minibatch $\mathcal{B}$ from $\mathcal{X}_l$.
Update θ by Eq. 4.1 using AdamW.
end for// *Conduct contrastive self-training with all data.*

for $t = 1, 2, \cdots, T_2$ **do** Update pseudo-labels $\widetilde{\boldsymbol{y}}$ by Eq. 4.3 for all $x \in \mathcal{X}$.

    for $k = 1, 2, \cdots, T_3$ **do** Sample a minibatch $\mathcal{B}$ from $\mathcal{X}$.
Select high confidence samples $\mathcal{C}$ by Eq. 4.9.
Calculate $\mathcal{L}_c$ by Eq. 4.10, $\mathcal{R}_1$ by Eq. 4.6, $\mathcal{R}_2$ by Eq. 4.12, and $\mathcal{L}$ by Eq. 4.4.
Update θ using AdamW.
    end for
end for

Output: Fine-tuned model $f(\cdot;\theta)$.

model as

$$\widetilde{\boldsymbol{y}}_j = \frac{[f(x;\theta)]_j^2 / f_j}{\sum_{j' \in \mathcal{Y}} [f(x';\theta)]_{j'}^2 / f_{j'}}, \tag{4.3}$$

where $f_j = \sum_{x' \in \mathcal{B}} [f(x';\theta)]_j^2$ is the sum over soft frequencies of class j. The non-binary soft pseudo-labels guarantee that, even if our prediction is inaccurate, the error propagated to the model update step will be smaller than using hard pseudo-labels.

$\diamond$ **Update θ with the current $\widetilde{\boldsymbol{y}}$.** We update the model parameters θ by minimizing

$$\mathcal{L}(\theta; \widetilde{\boldsymbol{y}}) = \mathcal{L}_c(\theta; \widetilde{\boldsymbol{y}}) + \mathcal{R}_1(\theta; \widetilde{\boldsymbol{y}}) + \lambda \mathcal{R}_2(\theta), \tag{4.4}$$

where $\mathcal{L}_c$ is the classification loss (Sec. 4.3.3), $\mathcal{R}_1(\theta; \widetilde{\boldsymbol{y}})$ is the contrastive regularizer (Sec. 4.3.2), $\mathcal{R}_2(\theta)$ is the confidence regularizer (Sec. 4.3.3), and λ is the hyper-parameter for the regularization.

4.3.2 Contrastive Learning on Sample Pairs

The key ingredient of our contrastive self-training method is to learn representations that encourage data within the same class to have similar representations and keep data in dif-

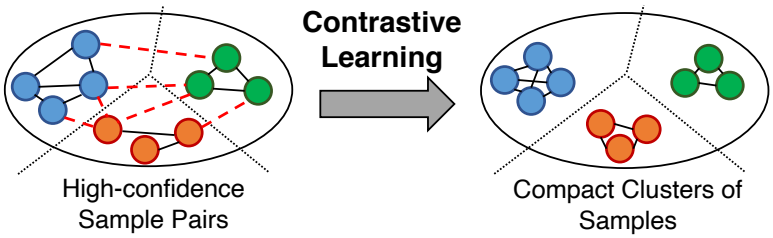


Figure 4.2: An illustration of contrastive learning. The black solid lines indicate similar sample pairs, and the red dashed lines indicate dissimilar pairs.

ferent classes separated. Specifically, we first select high-confidence samples (Sec. 4.3.3) $\mathcal{C}$ from $\mathcal{X}$. Then for each pair $x_i, x_j \in \mathcal{C}$, we define their similarity as

$$W_{ij} = \begin{cases} 1, & \text{if } \underset{k\in\mathcal{Y}}{\text{argmax}}[\widetilde{\boldsymbol{y}}_i]_k = \underset{k\in\mathcal{Y}}{\text{argmax}}[\widetilde{\boldsymbol{y}}_j]_k \\ 0, & \text{otherwise} \end{cases}, \tag{4.5}$$

where $\widetilde{\boldsymbol{y}}_i$, $\widetilde{\boldsymbol{y}}_j$ are the soft pseudo-labels (Eq. 4.3) for x_i, x_j, respectively. For each $x \in \mathcal{C}$, we calculate its representation $\boldsymbol{v} = \text{BERT}(x) \in \mathbb{R}^d$, then we define the contrastive regularizer as

$$\mathcal{R}_1(\theta; \widetilde{\boldsymbol{y}}) = \sum_{(x_i,x_j)\in\mathcal{C}\times\mathcal{C}} \ell(\boldsymbol{v}_i, \boldsymbol{v}_j, W_{ij}), \tag{4.6}$$

where

$$\ell = W_{ij} d_{ij}^2 + (1 - W_{ij})[\max(0, \gamma - d_{ij})]^2. \tag{4.7}$$

Here, $\ell(\cdot,\cdot,\cdot)$ is the contrastive loss [144, 145], d_{ij} is the distance[4] between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, and γ is a pre-defined margin.

For samples from the same class, i.e., $W_{ij} = 1$, Eq. 4.6 penalizes the distance between them, and for samples from different classes, the contrastive loss is large if their distance is small. In this way, the regularizer enforces similar samples to be close, while keeping dissimilar samples apart by at least γ. Figure 4.2 illustrates the contrastive representations. We can see that our method produces clear inter-class boundaries and small intra-class

[4]We use scaled Euclidean distance $d_{ij} = \frac{1}{d}\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2$ by default. More discussions on W_{ij} and d_{ij} are in Appendix C.6.

distances, which eases the classification tasks.

4.3.3 Confidence-based Sample Reweighting and Regularization

While contrastive representations yield better decision boundaries, they require samples with high-quality pseudo-labels. In this section, we introduce reweighting and regularization methods to suppress error propagation and refine pseudo-label qualities.

Sample Reweighting.

In the classification task, samples with high prediction confidence are more likely to be classified correctly than those with low confidence. Therefore, we further reduce label noise propagation by a confidence-based sample reweighting scheme. For each sample x with the soft pseudo-label $\widetilde{\boldsymbol{y}}$, we assign x with a weight $\omega(x)$ defined by

$$\omega = 1 - \frac{H\left(\widetilde{\boldsymbol{y}}\right)}{\log(C)},\ H(\widetilde{\boldsymbol{y}}) = -\sum_{i=1}^{C} \widetilde{\boldsymbol{y}}_i \log \widetilde{\boldsymbol{y}}_i, \tag{4.8}$$

where $0 \leq H(\widetilde{\boldsymbol{y}}) \leq \log(C)$ is the entropy of $\widetilde{\boldsymbol{y}}$. Notice that if the prediction confidence is low, then $H(\widetilde{\boldsymbol{y}})$ will be large, and the sample weight $\omega(x)$ will be small, and vice versa. We use a pre-defined threshold ξ to select high confidence samples $\mathcal{C}$ from each batch $\mathcal{B}$ as

$$\mathcal{C} = \{x \in \mathcal{B} \mid \omega(x) \geq \xi\}. \tag{4.9}$$

Then we define the loss function as

$$\mathcal{L}_c(\theta, \widetilde{\boldsymbol{y}}) = \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \omega(x) \mathcal{D}_{\mathrm{KL}}\left(\widetilde{\boldsymbol{y}} \| f(x;\theta)\right), \tag{4.10}$$

where

$$\mathcal{D}_{\mathrm{KL}}(P\|Q) = \sum_k p_k \log \frac{p_k}{q_k} \tag{4.11}$$

is the Kullback–Leibler (KL) divergence.

Confidence regularization The sample reweighting approach promotes high confidence

Table 4.1: Dataset statistics. Here C is the number of classes, Cover (in %) is the fraction of instances covered by weak supervision sources in the training set, and Acc. (in %) is the precision of weak supervision.

Dataset	Task	C	#Train	#Dev	#Test	Cover	Acc.
AGNews	Topic	4	96k	12k	12k	56.4	83.1
IMDB	Sentiment	2	20k	2.5k	2.5k	87.5	74.5
Yelp	Sentiment	2	30.4k	3.8k	3.8k	82.8	71.5
MIT-R	Slot Filling	9	6.6k	1.0k	1.5k	13.5	80.7
TREC	Question	6	4.8k	0.6k	0.6k	95.0	63.8
Chemprot	Relation	10	12.6k	1.6k	1.6k	85.9	46.5
WiC	WSD	2	5.4k	0.6k	1.4k	63.4	58.8

samples during contrastive self-training. However, this strategy relies on wrongly-labeled samples to have low confidence, which may not be true unless we prevent over-confident predictions. To this end, we propose a confidence-based regularizer that encourages smoothness over predictions, defined as

$$\mathcal{R}_2(\theta) = \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \mathcal{D}_{\text{KL}} \left(\mathbf{u} \| f(x; \theta) \right), \tag{4.12}$$

where $\mathcal{D}_{\text{KL}}$ is the KL-divergence and $\mathbf{u}_i = 1/C$ for $i = 1, 2, \cdots, C$. Such term constitutes a regularization to prevent over-confident predictions and leads to better generalization [146].

4.4 Experiments

Datasets and Tasks. We conduct experiments on 6 NLP classification tasks using 7 public benchmarks: *AGNews* [147] is a Topic Classification task; *IMDB* [148] and *Yelp* [138] are Sentiment Analysis tasks; *TREC* [149] is a Question Classification task; *MIT-R* [150] is a Slot Filling task; *Chemprot* [151] is a Relation Classification task; and *WiC* [152] is a Word Sense Disambiguation (WSD) task. The dataset statistics are summarized in Table 4.1. More details on datasets and weak supervision sources are in Appendix C.1.

Baselines. We compare our model with different groups of baseline methods:

(i) **Exact Matching (ExMatch)**: The test set is directly labeled by weak supervision

sources.

(ii) **Fine-tuning Methods**: The second group of baselines are fine-tuning methods for LMs:

◇*RoBERTa* [16] uses the RoBERTa-base model with task-specific classification heads.

◇ *Self-ensemble* [25] uses self-ensemble and distillation to improve performances.

◇ *FreeLB* [26] adopts adversarial training to enforce smooth outputs.

◇ *Mixup* [153] creates virtual training samples by linear interpolations.

◇ *SMART* [27] adds adversarial and smoothness constraints to fine-tune LMs and achieves state-of-the-art result for many NLP tasks.

(iii) **Weakly-supervised Models**: The third group of baselines are weakly-supervised models[5]:

◇ *Snorkel* [28] aggregates different labeling functions based on their correlations.

◇ *WeSTClass* [138] trains a classifier with generated pseudo-documents and use self-training to bootstrap over all samples.

◇ *ImplyLoss* [24] co-trains a rule-based classifier and a neural classifier to denoise.

◇ *Denoise* [34] uses attention network to estimate reliability of weak supervisions, and then reduces the noise by aggregating weak labels.

◇ *UST* [154] is state-of-the-art for self-training with limited labels. It estimates uncertainties via MC-dropout [155], and then select samples with low uncertainties for self-training.

Evaluation Metrics. We use classification accuracy on the test set as the evaluation metric for all datasets except MIT-R. MIT-R contains a large number of tokens that are labeled as "Others". We use the micro F_1 score for this dataset.[6]

Auxiliary. We implement COSINE using PyTorch[7], and we use RoBERTa-base as the pre-trained LM. Datasets and weak supervision details are in Appendix C.1. Baseline settings are in Appendices C.2 and C.3. Training details and setups are in Appendix C.4. Discussions on early-stopping are in Appendix C.5. Comparison of distance metrics and

[5]All methods use RoBERTa-base as the backbone unless otherwise specified.

[6]The Chemprot dataset also contains "Others" type, but such instances are few, so we still use accuracy as the metric.

[7]https://pytorch.org/

similarity measures are in Appendix C.6. Variance and significance tests are reported in Appendix C.7.

4.4.1 Learning From Weak Labels

Table 4.2: Classification accuracy (in %) on various datasets. We report the mean over three runs.

Method	**AGNews**	**IMDB**	**Yelp**	**MIT-R**	**TREC**	**Chemprot**	**WiC (dev)**
ExMatch	52.31	71.28	68.68	34.93	60.80	46.52	58.80
Fully-supervised Result							
RoBERTa-CL$^{\diamond}$ [16]	91.41	94.26	97.27	88.51	96.68	79.65	70.53
Baselines							
RoBERTa-WL† [16]	82.25	72.60	74.89	70.95	62.25	44.80	59.36
Self-ensemble [25]	85.72	86.72	80.08	72.88	66.18	44.62	62.71
FreeLB [26]	85.12	88.04	85.68	73.04	67.33	45.68	63.45
Mixup [153]	85.40	86.92	92.05	73.68	66.83	51.59	64.88
SMART [27]	86.12	86.98	88.58	73.66	68.17	48.26	63.55
Snorkel [28]	62.91	73.22	69.21	20.63	58.60	37.50	$-^{*}$
WeSTClass [138]	82.78	77.40	76.86	$-^{\otimes}$	37.31	$-^{\otimes}$	48.59
ImplyLoss [24]	68.50	63.85	76.29	74.30	80.20	53.48	54.48
Denoise [34]	85.71	82.90	87.53	70.58	69.20	50.56	62.38
UST [154]	86.28	84.56	90.53	74.41	65.52	52.14	63.48
Our COSINE **Framework**							
Init	84.63	83.58	81.76	72.97	65.67	51.34	63.46
COSINE	**87.52**	**90.54**	**95.97**	**76.61**	**82.59**	**54.36**	**67.71**

$^{\diamond}$: RoBERTa is trained with clean labels. †: RoBERTa is trained with weak labels. *: unfair comparison. $^{\otimes}$: not applicable.

We summarize the weakly-supervised leaning results in Table 4.2. In all the datasets, COSINE outperforms all the baseline models. A special case is the WiC dataset, where we use WordNet[8] to generate weak labels. However, this enables Snorkel to access some labeled data in the development set, making it unfair to compete against other methods. We will discuss more about this dataset in Sec. 4.4.3.

In comparison with directly fine-tuning the pre-trained LMs with weakly-labeled data, our model employs an "earlier stopping" technique[9] so that it does not overfit on the label noise. As shown, indeed "Init" achieves better performance, and it serves as a good initialization for our framework. Other fine-tuning methods and weakly-supervised models either cannot harness the power of pre-trained language models, e.g., Snorkel, or rely on clean labels, e.g., other baselines. We highlight that although UST, the state-of-the-art method to date, achieves strong performance under few-shot settings, their approach cannot estimate confidence well with noisy labels, and this yields inferior performance. Our model can gradually correct wrong pseudo-labels and mitigate error propagation via contrastive self-training.

It is worth noticing that on some datasets, e.g., AGNews, IMDB, Yelp, and WiC, our model achieves the same level of performance with models (RoBERTa-CL) trained with clean labels. This makes COSINE appealing in the scenario where only weak supervision is available.

4.4.2 Robustness Against Label Noise

Our model is robust against excessive label noise. We corrupt certain percentage of labels by randomly changing each one of them to another class. This is a common scenario in crowd-sourcing, where we assume human annotators mis-label each sample with the same probability. Figure 4.3 summarizes experiment results on the TREC dataset. Compared

[8] https://wordnet.princeton.edu/

[9] We discuss this technique in Appendix C.5.

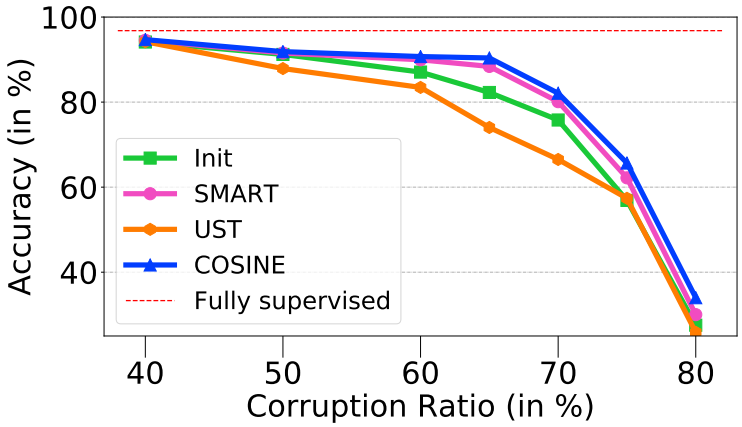


Figure 4.3: Results of label corruption on TREC. When the corruption ratio is less than 40%, the performance is close to the fully supervised method.

Table 4.3: Semi-supervised Learning on WiC. VAT (Virtual Adversarial Training) and MT (Mean Teacher) are semi-supervised methods. †: has access to weak labels.

Model	Dev	Test	#Params
Human Baseline	80.0		–
BERT [156]	–	69.6	335M
RoBERTa [16]	70.5	69.9	356M
T5 [99]	–	76.9	11,000M
Semi-Supervised Learning			
SenseBERT [157]	–	72.1	370M
RoBERTa-WL† [16]	72.3	70.2	125M
w/ MT† [65]	73.5	70.9	125M
w/ VAT† [59]	74.2	71.2	125M
w/ COSINE†	**76.0**	**73.2**	125M
Transductive Learning			
Snorkel† [28]	80.5	–	1M
RoBERTa-WL† [16]	81.3	76.8	125M
w/ MT† [65]	82.1	77.1	125M
w/ VAT† [59]	84.9	79.5	125M
w/ COSINE†	**89.5**	**85.3**	125M

with advanced fine-tuning and self-training methods (e.g. SMART and UST)[10], our model consistently outperforms the baselines.

[10]Note that some methods in Table 4.2, e.g., ImplyLoss and Denoise, are not applicable to this setting since they require weak supervision sources, but none exists in this setting.

4.4.3 Semi-supervised Learning

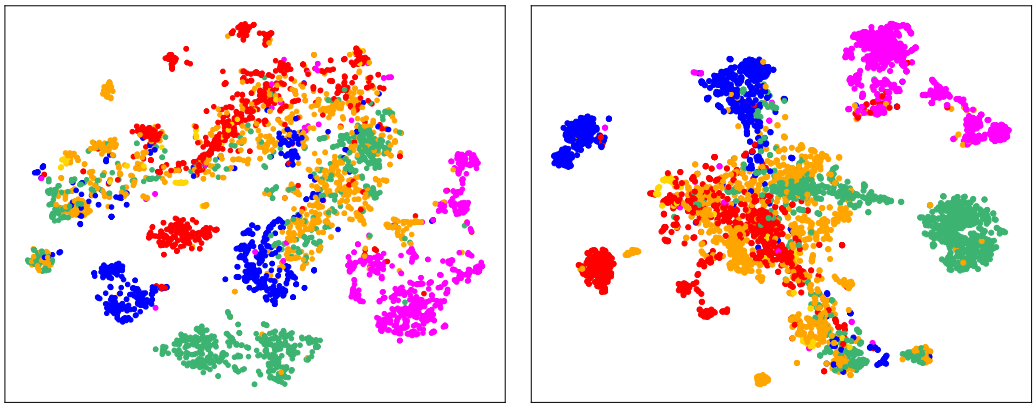
We can naturally extend our model to semi-supervised learning, where clean labels are available for a portion of the data. We conduct experiments on the WiC dataset. As a part of the SuperGLUE [158] benchmark, this dataset proposes a challenging task: models need to determine whether the same word in different sentences has the same sense (meaning).

Different from previous tasks where the labels in the training set are noisy, in this part, we utilize the clean labels provided by the WiC dataset. We further augment the original training data of WiC with unlabeled sentence pairs obtained from lexical databases (e.g., WordNet, Wictionary). Note that part of the unlabeled data can be weakly-labeled by rule matching. This essentially creates a *semi-supervised* task, where we have labeled data, weakly-labeled data and unlabeled data.

Since the weak labels of WiC are generated by WordNet and partially reveal the true label information, Snorkel [28] takes this unfair advantage by accessing the unlabeled sentences and weak labels of validation and test data. To make a fair comparison to Snorkel, we consider the transductive learning setting, where we are allowed access to the same information by integrating unlabeled validation and test data and their weak labels into the training set. As shown in Table 4.3, COSINE with transductive learning achieves better performance compared with Snorkel. Moreover, in comparison with semi-supervised baselines (i.e. VAT and MT) and fine-tuning methods with extra resources (i.e., SenseBERT), COSINE achieves better performance in both semi-supervised and transductive learning settings.

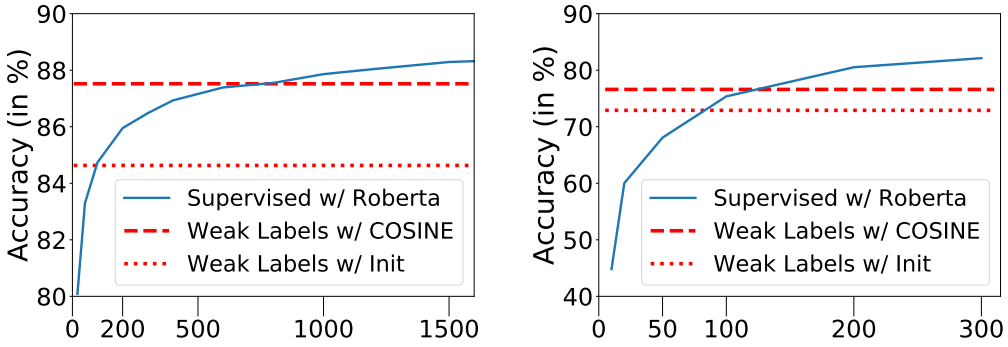
4.4.4 Case Study

Error propagation mitigation and wrong-label correction. Figure 4.7 visualizes this process. Before training, the semantic rules make noisy predictions. After the initialization step, model predictions are less noisy but more biased, e.g., many samples are mis-labeled as “Amenity”. These predictions are further refined by contrastive self-training. The right-



(a) Embedding w/o $\mathcal{R}_1$.] (b) Embedding w/ $\mathcal{R}_1$.

Figure 4.4: t-SNE [159] visualization on TREC. Each color denotes a different class.



(a) Results on Agnews.] (b) Results on MIT-R.

Figure 4.5: Accuracy vs. Number of annotated labels.

most figure demonstrates *wrong-label correction*. Samples are indicated by radii of the circle, and classification correctness is indicated by color, i.e., blue means correct and orange means incorrect. From inner to outer tori specify classification accuracy after the initialization stage, and the iteration 1,2,3. We can see that many incorrect predictions are corrected within three iterations. To illustrate: the right black dashed line means the corresponding sample is classified correctly after the first iteration, and the left dashed line indicates the case where the sample is mis-classified after the second iteration but corrected after the third. These results demonstrate that our model can correct wrong predictions via contrastive self-training.

Better data representations. We visualize sample embeddings in Fig. 4.4. By incorporat-

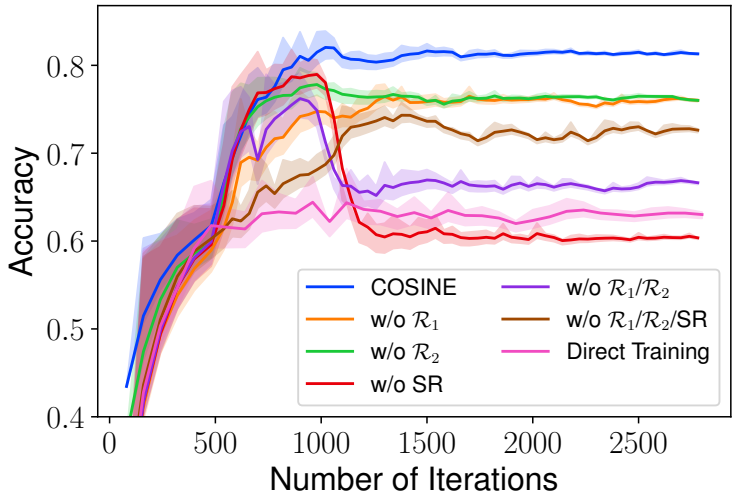


Figure 4.6: Learning curves on TREC with different settings. Mean and variance are calculated over 3 runs.

ing the contrastive regularizer $\mathcal{R}_1$, our model learns more compact representations for data in the same class, e.g., the green class, and also extends the inter-class distances, e.g., the purple class is more separable from other classes in Fig. 4.4b than in Fig. 4.4a.

Label efficiency. Figure 4.5 illustrates the number of clean labels needed for the supervised model to outperform COSINE. On both of the datasets, the supervised model requires a significant amount of clean labels (around 750 for Agnews and 120 for MIT-R) to reach the level of performance as ours, whereas our method assumes no clean sample.

Higher Confidence Indicates Better Accuracy. Figure 4.9 demonstrates the relation between prediction confidence and prediction accuracy on IMDB. We can see that in general, samples with higher prediction confidence yield higher prediction accuracy. With our sample reweighting method, we gradually filter out low-confidence samples and assign higher weights for others, which effectively mitigates error propagation.

4.4.5 Ablation Study

Components of COSINE. We inspect the importance of various components, including the contrastive regularizer $\mathcal{R}_1$, the confidence regularizer $\mathcal{R}_2$, and the sample reweighting (SR) method, and the soft labels. Table 4.4 summarizes the results and Fig. 4.6 visualizes

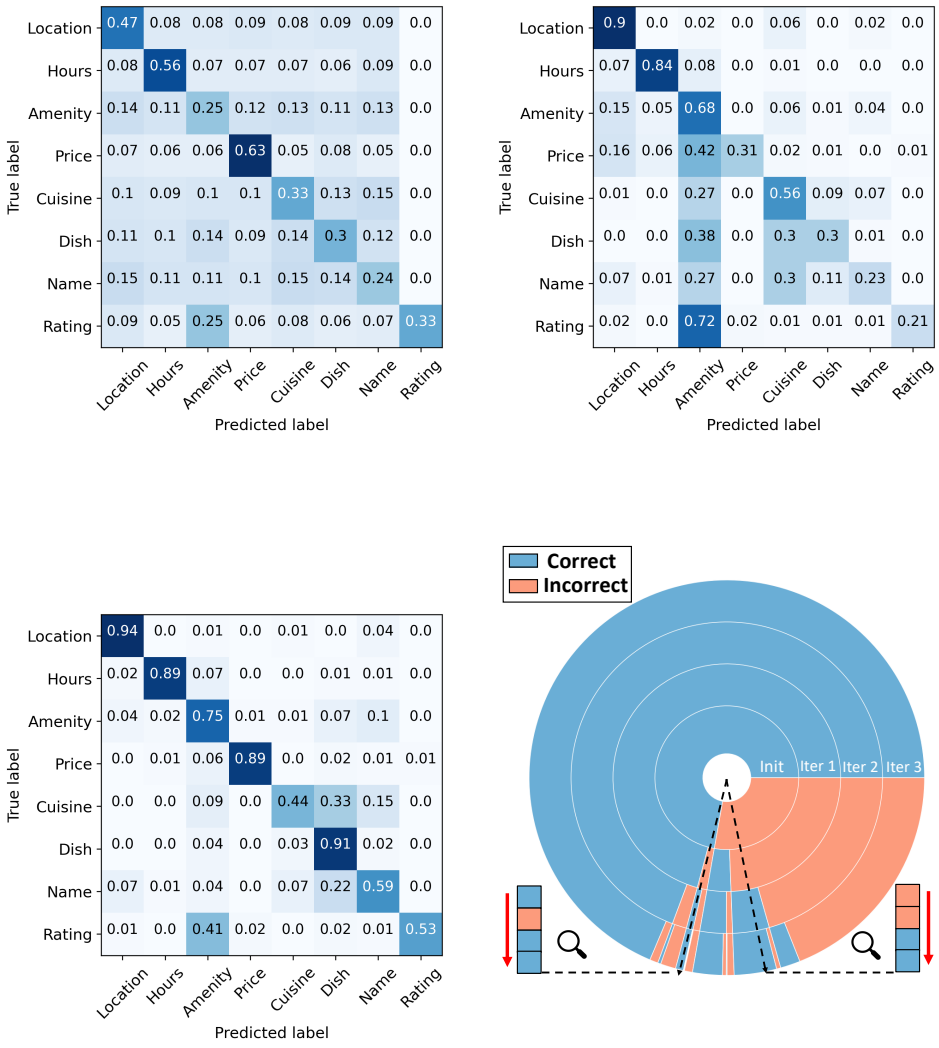
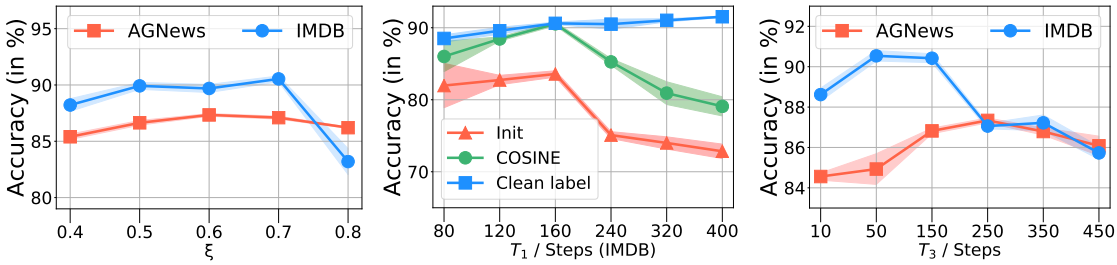


Figure 4.7: Classification performance on MIT-R. From top-left to button-right: visualization of ExMatch, results after the initialization step, results after contrastive self-training, and wrong-label correction after training.

the learning curves. We remark that all the components jointly contribute to the model performance, and removing any of them hurts the classification accuracy. For example, sample reweighting is an effective tool to reduce error propagation, and removing it causes the model to eventually overfit to the label noise, e.g., the red bottom line in Fig. 4.6 illustrates that the classification accuracy increases and then drops rapidly. On the other hand, replacing the soft pseudo-labels (Eq. 4.3) with the hard counterparts (Eq. 4.2) **causes drops** in performance. This is because hard pseudo-labels lose prediction confidence information.



(a) Effect of ξ. (b) Effect of T_1. (c) Effect of T_3.

Figure 4.8: Effects of different hyper-parameters.

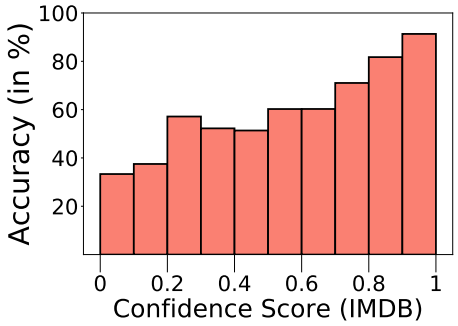


Figure 4.9: Accuracy vs. Confidence score.

Table 4.4: Effects of different components. Due to space limit we only show results for 5 representative datasets.

Method	**AGNews**	**IMDB**	**Yelp**	**MIT-R**	**TREC**
Init	84.63	83.58	81.76	72.97	66.50
COSINE	**87.52**	**90.54**	**95.97**	**76.61**	**82.59**
w/o $\mathcal{R}_1$	86.04	88.32	94.64	74.11	78.28
w/o $\mathcal{R}_2$	85.91	89.32	93.96	75.21	77.11
w/o SR	86.72	87.10	93.08	74.29	79.77
w/o $\mathcal{R}_1$/$\mathcal{R}_2$	86.33	84.44	92.34	73.67	76.95
w/o $\mathcal{R}_1$/$\mathcal{R}_2$/SR	86.61	83.98	82.57	73.59	74.96
w/o Soft Label	86.07	89.72	93.73	73.05	71.91

Hyper-parameters of COSINE. In Fig. 4.8, we examine the effects of different hyper-parameters, including the confidence threshold ξ (Eq. 4.9), the stopping time T_1 in the initialization step, and the update period T_3 for pseudo-labels. From Fig. 4.8a, we can see that setting the confidence threshold too big hurts model performance, which is because an over-conservative selection strategy can result in insufficient number of training data. The

stopping time T_1 has drastic effects on the model. This is because fine-tuning COSINE with weak labels for excessive steps causes the model to unavoidably overfit to the label noise, such that the contrastive self-training procedure cannot correct the error. Also, with the increment of T_3, the update period of pseudo-labels, model performance first increases and then decreases. This is because if we update pseudo-labels too frequently, the contrastive self-training procedure cannot fully suppress the label noise, and if the updates are too infrequent, the pseudo-labels cannot capture the updated information well.

4.5 Discussion and Related Works

Fine-tuning Pre-trained Language Models. To improve the model's generalization power during fine-tuning stage, several methods are proposed [20, 141, 26, 27, 25, 130, 160, 161, 162, 163]. However, as shown in experiments, these methods rely heavily on large amounts of *clean labels*, which are not always available. To address this issue, we propose a contrastive self-training framework that fine-tunes pre-trained models with only weak labels.

Learning From Weak Supervision. In weakly-supervised learning, the training data are usually noisy and incomplete. Existing methods aim to denoise the sample labels or the labeling functions by, for example, aggregating multiple weak supervisions [28, 164, 34], using clean samples [24], and leveraging contextual information [165]. However, most of them can only use specific type of weak supervision on specific task, e.g., keywords for text classification [143, 165], and they require prior knowledge on weak supervision sources [24, 164, 34]. Our work is orthogonal to them since we do not denoise the labeling functions directly. Instead, we adopt contrastive self-training to leverage the power of pre-trained language models for denoising, which is *task-agnostic* and applicable to various NLP tasks with minimal additional efforts.

4.6 Conclusion

In this chapter, we propose a contrastive regularized self-training framework, COSINE, for fine-tuning pre-trained language models with weak supervision. Our framework can learn better data representations to ease the classification task, and also efficiently reduce label noise propagation by confidence-based reweighting and regularization. We conduct experiments on various classification tasks, including sequence classification, token classification, and sentence pair classification, and the results demonstrate the efficacy of our model.

CHAPTER 5

AUTOMATIC DIALOGUE EVALUATION WITH OFF-POLICY EVALUATION

This chapter focuses on automatic evaluation for dialogue systems. The content is based on the following publication:

Haoming Jiang et al. (2021). "Towards Automatic Evaluation of Dialog Systems: A Model-Free Off-Policy Evaluation Approach". In: *arXiv preprint arXiv:2102.10242*

5.1 Overview

One of the fundamental research bottlenecks for developing dialog systems falls in evaluation, namely how to measure the performance of these systems in an automatic and scalable manner. Different from supervised natural language understanding tasks (e.g., text classification and machine translation), an ideal environment for evaluating dialog systems, also known as the Turing test, involves multi-turn human interaction [37, 38, 39, 40]. While online platforms such as Amazon Mechanical Turk can provide human-based evaluation, they are often expensive and not scalable[41].

Researchers have adopted language quality metrics for single-turn response generation given a fixed context (e.g., BLEU score and perplexity) to automatically evaluate dialog systems [42, 43, 44, 45, 41]. However, these metrics only weakly correlate to human evaluation in practice [38, 39]. One cause of such weak correlation is that language quality metrics rely on the exact match between generated text and ground-truth, which generally do not fully overlap. While certain embedding-based metrics have been developed to combat this lack of coverage [46, 47], they are only post-hoc judgments based on static experience data, and does not necessarily reflect the dynamic quality of multi-turn interactive dialog well [39]. Moreover, evaluation of goal-oriented dialog systems should be based on how well dialog systems collect information from users and whether the goal is completed;

language quality metrics are thus unable to meet these requirements.

To overcome the limitations of the aforementioned static evaluation methods, another line of work has proposed to model the interactive process of a conversation as a Markov decision process (MDP) [48, 49, 50, 51, 39, 52]. Accordingly, automatic evaluation of dialog systems can be formulated as an off-policy evaluation (OPE) problem, where a human subject is the so-called "environment" in the reinforcement learning (RL) literature. For instance, Wei et al. [4] propose a model-based approach for goal-orient dialog systems. They first learn an environment/human model from the experience data consisting of human response, and then evaluate a dialog agent/policy by executing the policy within the learned environment. This procedure is known as "self-play evaluation". Such a model-based approach requires accurate estimation of an environment/human when both input and output are in a *combinatorially* large space, i.e., the trained model needs to be able to mimic complex human behavior of generating meaningful sentences from huge vocabulary. Unfortunately, such a requirement is far beyond the current capability of model-based reinforcement learning algorithms. As a result, evaluations that rely on accurate modeling of the environment is often unreliable. A similar model-based approach is proposed [39] to evaluate open-domain chit-chat dialog systems. In addition to modeling human behavior, they also model the reward function (for mimicking the complex mechanism behind human ratings) based on handcrafted features, which makes evaluation even more unreliable.

In this chapter, we propose a general OPE framework named ENIGMA (EvaluatiNg dIaloG systeMs Automatically) for estimating human evaluation score (i.e., how a human would rate a dialog system). Different from the aforementioned model-based approaches, which rely on complex modeling of human behavior given combinatorially large vocabulary, ENIGMA takes advantage of recent advances in model-free OPE and avoids direct modeling of dynamic transitions and reward functions in a complex environment. Moreover, ENIGMA overcomes several limitations of existing OPE methods in order to evaluate dialog systems: **(I)** Existing OPE methods only apply to infinite or fixed horizon settings

(where horizon length corresponds to number of turns in a conversation), while onversations, on the other hand, often have varying horizon lengths; **(II)** Existing OPE methods require experience data to sufficiently cover states and actions a target policy might visit. Due to limited experience data and the combinatorial nature of languages, such a requirement can hardly be satisfied in dialog evaluation; **(III)** Certain OPE methods rely on accurate estimation of the behavior policies used to collect the experience data. Unfortunately, such behavior policies are humans or complex dialog systems, and estimating their probabilistic model is essentially a challenging imitation learning problem. [1]

To address **(I)**, we propose a pseudo state padding method, which augments each conversation into infinitely many turns and yet preserves the original policy value; to address **(II)**, we leverage pre-trained language models [8], which essentially transfer knowledge from out-of-domain data to alleviate the coverage issue; to address **(III)**, we adopt a stationary distribution correction estimation approach [56], which directly models the state-action density ratio between the experience data and the target policy [57], and is therefore agnostic to the behavior policy. We summarize ENIGMA in comparison to existing works in Table 5.1.[2]

We conduct thorough experiments on evaluating goal-oriented (AirDialog, Wei et al. [4]) and chit-chat (ConvAI2, Dinan et al. [58]) dialog systems to demonstrate the superiority of ENIGMA. Specifically, we follow the experimental settings similar to Ghandeharioun et al. [39] and See et al. [40] (See details in Section 5.4), and show ENIGMA significantly outperforms the existing static evaluation and self-play evaluation methods in both domains.

[1] Note that even though some of the model-free OPE estimators still require modeling behavior policies, they are still significantly easier than model-based OPE, which has to model the underlying dialog environment.

[2] We only present a compact table due to space limit. More details can be found in Appendix D.5.

Table 5.1: Comparison of existing works on Automatic Evaluation of Dialog Systems.

Method	Criterion	Dynamic	Model-Free	Behavior-Policy
BLEU, PPL	Language Quality	No	N/A	Human
Lowe et al. [41]	Language Quality	No	N/A	Model & Human
Wei et al. [4]	Task Completion	Yes	No	Human
Ghandeharioun et al. [39]	Language Score	Yes	No	Model
ENIGMA	Both	Yes	Yes	Model

5.2 Background

• **Dialog Generation as Markov Decision Process**. A conversation is generated through interactions alternating between an agent π (i.e., a dialog system) and an environment $\mathcal{E}$ (i.e., a human). We denote the dialog as $h = \{e_0, a_1, e_1, ..., a_T\}$, where a_i and e_i are sentences generated by π and $\mathcal{E}$ respectively, and T is the number of turns in the conversation. Dialog can be naturally described as a Markov decision process (MDP) [167] $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \mu_0 \rangle$. Specifically, at the t-th turn, state $s_t \in \mathcal{S}$ captures the previous conversation history $s_t = \{e_0, a_1, e_1, ..., a_{t-1}, e_{t-1}\}$. An action $a_t \in \mathcal{A}$ is an agent's response given this context. Conversation can then be represented by the last state and action, i.e., $h = \{s_T, a_T\}$. An agent π is essentially a policy that maps $\mathcal{S}$ to $\mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\cdot)$ denotes the set of probability measures over the action space. A transition kernel $P(\cdot|s_t, a_t)$ returns s_{t+1} as the state at turn $t+1$, and an environment $\mathcal{E}$ generates a reward $r_t = R(s_t, a_t) \in [0, 1]$. Note that s_{t+1} essentially concatenates s_t and a_t with e_t, where e_t is a response from the environment (i.e., human) at the t-th turn. The initial state $s_1 = \{e_0\}$ is randomly sampled from some distribution μ_0. An illustrative example of the dialog on booking a flight ticket is shown in Figure 5.1 [4].

Note that the reward $r_t = R(s_t, a_t)$ generated by the environment depends on the purpose of the dialog system: for open-domain chit-chat dialog, R measures language quality; for goal-oriented agents, R measures task completion scores. In particular, we follow the *sparse reward* setting, where each conversation is only evaluated at the ending state, i.e., $r_t = 0$ for $t < T$ [4].

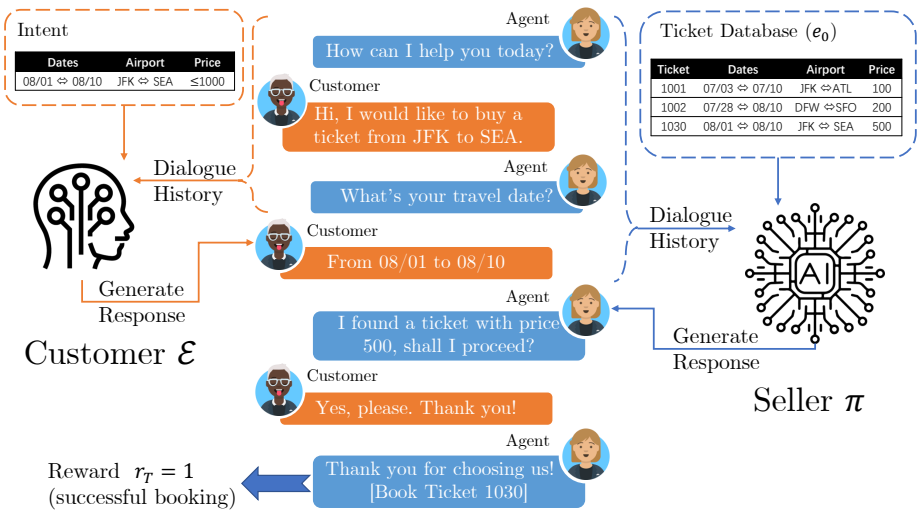


Figure 5.1: Dialog for booking a flight ticket (Airdialog).

• **Automatic Dialog Evaluation as Off-Policy Evaluation**. Dialog evaluation can be naturally viewed as computing the expected reward of the above MDP defined as

$$\rho(\pi) = \mathbb{E}_{h\sim\mu_0,\pi,\mathcal{E}}[R(s_T, a_T)], \tag{5.1}$$

where $h = \{s_T, a_T\}$ is sampled from the initial distribution μ_0 and the interaction between π and $\mathcal{E}$. When the environment (i.e., human) is accessible, $\rho(\pi)$ can be directly estimated by interaction with the environment, which is known as *on-policy evaluation* [168].

In dialog systems, however, interaction with human is expensive or prohibitive in practice, so human-free automatic evaluation is desired. *Off-policy evaluation* (OPE) [169] is an appealing choice when access to the environment is limited or unavailable. In particular, OPE can estimate $\rho(\pi)$ based solely on pre-collected tuples $\{(s, a, r, s')_i\}_{i=1}^N$ from behavior policies that are different from π.

OPE has been considered as one of the most fundamental problems in RL. A straightforward approach is to first directly learn an environment model (R and P) from experience data and then estimate $\rho(\pi)$ by executing the policy within the learned environment. Such *model-based* OPE exactly corresponds to the so-called "self-play evaluation" in the dialog system literature [4, 39]. Unfortunately, it is notoriously difficult to specify a proper

model for highly complicated environments such as a dialog environment (i.e., a human), where the state and action spaces are combinatorially large due to huge vocabulary size and complex transitions. As a result, the estimation error of the environment accumulates as interaction proceeds, and model-based self-play evaluation of dialog systems often becomes unreliable [170].

To address the challenge above, many *model-free* OPE methods that avoid direct modeling of the environment have been proposed. Model-free OPE can be categorized into *behavior-aware* and *behavior-agnostic* methods. Specifically, behavior-aware methods rely on either knowing or accurately estimating the probabilistic model of behavior policies used for collecting the experience data (e.g., inverse propensity scoring, Horvitz and Thompson [171]). Unfortunately, behavior policies are often unknown in practice. Estimating their probabilistic models is also quite challenging, as it requires modeling human behaviors or complex dialog systems. Behavior-agnostic methods, on the other hand, do not require explicit knowledge or direct modeling of behavior policies, and are therefore more favorable when experience data is collected by multiple (potentially unknown) behavior policies.

Unfortunately, most of the existing model-free behavior-agnostic OPE methods focus on either infinite-horizon [56, 172, 173] or fixed-horizon settings [174, 175], and cannot be applied to evaluating dialog systems whose horizon (number of turns) vary between conversations. While LSTDQ [176] can be adopted to handle varying horizons, it has been shown to not work well under the sparse reward setting [176, 177].

5.3 ENGIMA

We present the ENIGMA framework for automatically evaluating dialog systems using experience data. In particular, ENIGMA is model-free and agnostic to behavior policies for generating the experience data. ENIGMA has three major ingredients: **(1)** pseudo-state padding for converting a dialog into an infinite-horizon MDP, **(2)** distribution-correction estimation (DICE, Nachum et al. [56]) with post-normalization for estimating the value of

the target policy based on experience data, and **(3)** function approximation and representation learning with pre-trained language models.

5.3.1 Pseudo-State Padding

As mentioned in Section 2, existing model-free behavior-agnostic OPE methods cannot handle varying horizon lengths in conversations under the sparse reward setting. To address this issue, we design a special padding scheme, so that the policy value can be estimated by OPE methods from the resulting padded MDP. We first pad conversation sequences with pseudo states, which leads to a padded MDP with a fixed horizon length $T_{\max}$. We then convert such a fixed horizon MDP into infinite horizon by augmentation, i.e., we repeatedly concatenate the ending state of the fixed horizon MDP to its initial state.

More specifically, as illustrated in Figures 5.2, the policy takes a deterministic action at all pseudo states, i.e., $\pi(a = \text{NextPad}|s = \text{Pad}_k) = 1$. The transition kernel of the new process can be defined as

$$
\begin{aligned}
&\text{Conversation Transition :}\\
&P(s' = s \cup a \cup e|s, a, \text{incomplete conv.}) = \mathcal{E}(e|s, a),\\
&\text{Jump into Pseudo States :}\\
&P(s' = \text{Pad}_{T+1}|s, a, \text{complete conv. with } T \text{ turns})=1,\\
&\text{Jump between Pseudo States :}\\
&P(s' = \text{Pad}_{k+1}|s = \text{Pad}_k, a = \text{NextPad}, k < T_{\max})=1,\\
&\text{Jump out of Pseudo States :}\\
&P(s'|s = \text{Pad}_{T_{\max}}, a = \text{NextPad}) = \mu_0\left(s'\right).
\end{aligned}
$$

This new process is still a valid MDP, as its transition kernel satisfies the Markov property. For notational simplicity, we refer to such an augmented MDP with infinite horizon

as "the augmented MDP".

Accordingly, the policy value of π for the augmented MDP can be defined as

$$\rho_A(\pi) = \lim_{N\to\infty} \mathbb{E}_{(h_1,h_2,\ldots,h_N)\sim\mu_0,\pi,\mathcal{E}}\left[\frac{1}{NT_{\max}} \sum_{i=1}^{N} \sum_{t=1}^{T_{\max}} R(s_t^{(i)}, a_t^{(i)})\right], \tag{5.2}$$

where h_i's are padded conversations sampled from interactions between π and $\mathcal{E}$. Since there is only one non-zero reward for every $T_{\max}$ steps, rewards in the augmented MDP are also sparse.

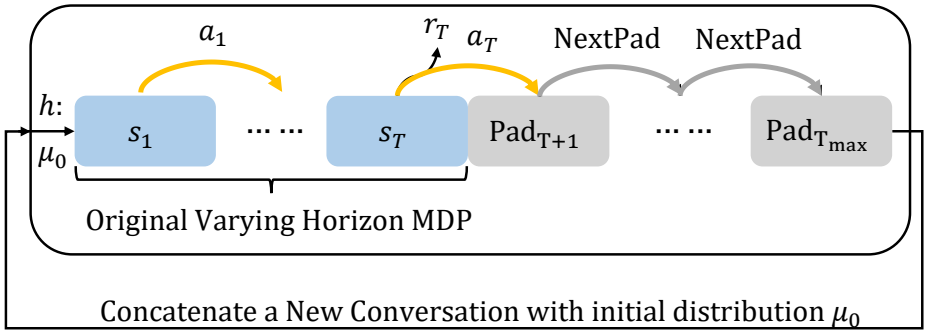


Figure 5.2: Augmented MDP with Infinite Horizon.

We justify such a padding scheme in the following theorem showing that the augmented MDP has a unique stationary distribution, and guarantees any policy π to have a finite policy value. Moreover, the policy value of π under the augmented MDP is proportional to its counterpart under the original MDP without augmentation. Due to space limit, we defer the proof to Appendix D.1.1.

Theorem 1. The augmented MDP with infinite horizon satisfies the following properties:

• It has a unique stationary state-action visitation distribution $d^\pi(s, a)$;

• For the station-action pair (s_t, a_t) in a conversation h with padded pseudo states, we have

$$\begin{aligned} d^\pi(s_t, a_t) =& \frac{1}{T_{\max}} \sum_{\{(s_k,a_k)\}_{k=1}^{t-1}} [\mu_0(s_1)\pi(a_1|s_1) \\ & P(s_2|a_1, s_1) \cdots P(s_t|a_{t-1}, s_{t-1})\pi(a_t|s_t)], \end{aligned} \tag{5.3}$$

where $\{(s_k, a_k)\}_{k=1}^{t-1}$ are the state-action pairs in the same conversation as (s_t, a_t);

• The policy value can be computed by sampling from $d^\pi(s, a)$, and we have

$$\rho_A(\pi) = \mathbb{E}_{(s,a)\sim d^\pi(s,a)}[R(s, a)] = \rho(\pi)/T_{\max}. \tag{5.4}$$

Remark 4. Some OPE methods, e.g., LSTDQ [176], can handle fixed horizons, therefore only applying the fixed-horizon padding would suffice. DICE estimators [56], on the other hand, can only handle infinite horizons, therefore the infinite-horizon augmentation is necessary.

Remark 5. Note that in practice, we do not actually need to concatenate infinitely many conversations for computing $\rho_A(\pi)$. As suggested by (5.4), $\rho_A(\pi)$ can be computed based on $d^\pi(s_t, a_t)$ defined in (5.3), which is the product of only finite terms.

5.3.2 Model-Free Behavior-Agnostic DICE Estimator

With the proposed augmentation, we obtain an infinite horizon MDP from which the policy value of the original MDP can be recovered. We then apply DICE [56, 173] to estimate $\rho_A(\pi)$ based on pre-collected experience data $\mathcal{D} = \{(s, a, r, s')_i\}_{i=1}^N$ without interacting with $\mathcal{E}$ (i.e., a human), where $(s, a) \sim d^\mathcal{D}$ are samples from some unknown distribution $d^\mathcal{D}$. We slightly abuse the notations and use $(s, a, r, s') \sim d^\mathcal{D}$ as a shorthand for $(s, a) \sim d^\mathcal{D}, r = R(s, a), s' \sim P(\cdot|s, a)$, which simulates sampling form the dataset $\mathcal{D}$.

DICE is a model-free policy evaluation method (without explicitly modeling $\mathcal{E}$) and does not require knowledge of behavior policies for generating the experience data, which provides a more reliable estimation of $\rho_A(\pi)$ than other OPE methods. Specifically, DICE decomposes $\rho_A(\pi)$ into:

$$\rho_A(\pi) = \mathbb{E}_{(s,a,r)\sim d^\mathcal{D}}[\zeta(s, a)r], \tag{5.5}$$

where $\zeta(s, a) := d^\pi(s, a)/d^\mathcal{D}(s, a)$ is the *distribution correction ratio*. Then DICE esti-

mates ζ by solving the following regularized minimax optimization problem:

$$\max_{\zeta \geq 0} \min_{\nu,\lambda} L_D(\zeta, \nu, \lambda) = \mathbb{E}_{(s,a,r,s') \sim d^{\mathcal{D}}, a' \sim \pi(s')}[\zeta(s,a) \cdot (\nu(s',a') - \nu(s,a))] + \mathbb{E}_{(s,a) \sim d^{\mathcal{D}}}[\lambda(\zeta(s,a) - 1)] - \alpha_\zeta \cdot \mathbb{E}_{(s,a) \sim d^{\mathcal{D}}}[f(\zeta(s,a))]. \tag{5.6}$$

where $\nu(s,a)$'s are auxiliary variables, f is a convex regularizer (e.g., $f(x) = x^2$), and α_ζ is a tuning parameter. Due to the space limit, we omit the details of deriving the DICE estimator. Please refer to Yang et al. [173] for more technical details.

• **Post-Normalization**. Note that (5.6) handles the constraint $\mathbb{E}_{(s,a) \sim d^{\mathcal{D}}} \zeta(s,a) = 1$ by Lagrange multipliers λ, which cannot guarantee that the constraint is *exactly* satisfied when solving (5.6) using alternating SGD-type algorithms [178, 179]. To address this issue, we propose a post-normalization step that explicitly enforces the constraint:

$$\rho_n(\pi) = \sum_{(s,a,r) \sim d^{\mathcal{D}}} \zeta(s,a) r \Big/ \sum_{(s,a) \sim d^{\mathcal{D}}} \zeta(s,a). \tag{5.7}$$

As we will see in our experiments in Section 5.4, the post-normalization step is crucial for DICE to attain good estimation accuracy in practice; without the post-normalization, we observe potential divergence in terms of policy value estimation.

• **Why do we prefer DICE?** Deep Q-learning and its variants are another popular model-free and behavior-agnostic approach to off-policy evaluation. However, due to the sparse rewards in dialogs, fitting the state-action value function (i.e., the Q-function) in deep Q-learning is notoriously difficult [177]. We observe in Section 5.4 that deep Q-learning is computationally unstable.

In contrast, DICE only needs to estimate the density correction ratio ζ, which is decoupled from the rewards associated with the policy value as shown from (5.6). This significantly alleviates the computational challenge incurred by sparse rewards. Moreover, DICE

also applies the post-normalization, additional regularization (i.e., $\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}[f(\zeta(s,a))]$), and constraints on ζ (i.e., $\zeta \geq 0$ and $\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}[\zeta(s,a)] = 1$), all of which further stabilize training. These features allow DICE achieve better estimation performance than deep Q-learning in dialog systems evaluation.

Recent progresses in OPE based on density ratio estimation are remarkable [57, 56, 180, 181], however, there exists a statistical limit in off-policy evaluation. Specifically, the Cramer-Rao lower bound of the MSE has been established in Jiang and Li [182], which is proportional to the square of the density ratio. This implies that we can only obtain accurate estimation of policy value only if the ratio ζ is not too large. While the ratio-based minimax algorithms should have achieved the asymptotic lower bound [183, 174], even better estimation results can be obtained when behavior and target policies are more similar. We thus introduce an experience data collection protocol in Section 5.4.1 which satisfies the bounded ratio requirement and ensures the success of OPE methods.

5.3.3 Function Approximation with RoBERTa

Despite the apparent advantages of DICE estimators, directly training DICE from scratch will fall short due to the bounded ratio requirement being quickly broken in the large combinatorial state-action space in dialog.

We alleviate this issue by learning reliable representations from an enormous amount of pre-collected data. We resort to the domain transfer learning technique, also known as language model pre-trained and fine-tuning [8]. For example, RoBERTa[16] is an extremely large bidirectional transformer model [9] pre-trained using huge amounts of open-domain text data in a self-supervised/unsupervised manner. RoBERTa is particularly attractive to the dialog evaluation task due to the following merits: (1) the pre-training process does not require any labelled data; (2) the pre-trained models are publicly available; (3) the massive model sizes (usually with hundreds of millions or billions of parameters) allow these models to effectively capture rich semantic and syntactic information of natural language

(rather than enumerating the original combinatorial language space).

To transfer the knowledge from the pre-trained RoBERTa model to dialog evaluation, we parameterize ζ and ν as follows: We keep the pre-trained RoBERTa encoder layer and replace the original mask language modeling head by a two-layer fully connected network with a scalar output. For simplicity, we denote the corresponding parametric forms of ζ and ν as RoBERTa-ζ and RoBERTa-ν, respectively. Note that we only need RoBERTa-ζ and RoBERTa-ν to share the same encoder, as illustrated in Figure 5.3. We then use RoBERTa-ζ and RoBERTa-ν as the initial solution to solve (5.6), which is also known as the fine-tuning step [8, 16].

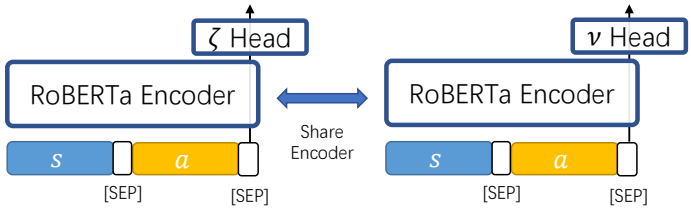


Figure 5.3: Function Approximation with RoBERTa.

With a properly designed mask, the self-attention mechanism in the bi-direction transformer architecture allows us to efficiently compute $\zeta(s, a)$ and $\nu(s, a)$ for all state-action pairs in the same dialog simultaneously. Due to the space limit, we defer the mask design details to Appendix D.1.2.

5.4 Experiments

We empirically evaluate ENIGMA on two dialog datasets: AirDialog [4] for goal-oriented tasks and ConvAI2 [58] for open-domain chit-chat respectively. See details of experimental setup in Appendix D.2. [3]

[3] We release our source code for ENIGMA algorithm on Github: https://github.com/google-research/google-research/tree/master/dialogue_ope/airdialogue_ope. Transformer model for AirDialog: https://github.com/google-research/google-research/tree/master/dialogue_ope/airdialogue_model_transformer. Experience data for OPE on ConvAI2: http://hmjianggatech.github.io/files/data/convai2_opedata.zip.

5.4.1 Policy Training Data and Experience Data

As mentioned in Section 5.3.2, there exists an information theoretic limit for all off-policy evaluation methods: no method can perform well when the state-action density ratio between the target and behavior policy is too large. To avoid such a circumstance, we need to ensure that the experience data collected by a behavior policy do not deviate too much from data induced by the target policy. Unfortunately, both datasets used in our experiments do not satisfy such a requirement. AirDialog, for example, consists of dialog between humans, which are near-perfect golden samples as human agents almost always successfully book tickets for customers. Dialog system agents, on the other hand, have many failure modes (i.e., the target policy/agent does not book the correct ticket for a human customer). Hence, directly using human dialog as the behavior data to evaluate dialog agents is subjected to limitations.

In order to properly evaluate an imperfect target policy in the presence of the information theoretic limit, we refer to Lowe et al. [41], Ghandeharioun et al. [39], and See et al. [40], and collect experience data using behavior policies similar to the target policy. To avoid confusion, we call data collected by the behavior policy "experience data" and data used to train an agent "policy training data". More details are elaborated below for each dataset.

It is worth noting that existing work on dialog systems evaluation also enforces similar requirements. For example, Lowe et al. [41] show higher Pearson correlation coefficient (0.37) between automatic metrics and human ratings when behavior policies contain the target policy. When the target policy is excluded from behavior policies, however, the correlation is only 0.13, even lower than the meaningless correlation between dialog lengths and human ratings (0.27). Another example is Ghandeharioun et al. [39], where the studied agents are similar to each other in their hierarchical architectures, hyperparameters, and training data.

We compare the experience data used in this chapter with existing works in Table 5.2.

Table 5.2: Number of Dialogues/Agents in Experience Data. †: 4,104 dialog turns and the number of dialogs is not avaliable.

Experience Data	Agents	Dialogs
Lowe et al. [41]	4	N/A †
Ghandeharioun et al. [39]	12	512
Ours-Airdialog	24	2400
Ours-ConvAI2	29	2616

5.4.2 Goal-Oriented Systems

We first test ENIGMA for evaluating goal-oriented dialog systems on a flight ticket booking task.

• **Policy Training Data**. We use the ***AirDialog*** dataset[4] for policy training [4]. It contains 402,038 pieces of dialog from human sellers and human customers collaborating on buying flight tickets. We use different proportions of the dataset and different hyperparameters to train 24 seller agents using behavior cloning (See Appendix D.3 for details) [5].

• **Experience Data**. We invite 20 people to evaluate the 24 seller agents. Specifically, each of the 20 human customers interacts with a seller agent 5 times to generate 100 pieces of dialog, and gives each piece an evaluation score between 0 and 1. The final score an agent receives is the average of the 100 scores. We consider three types of scores: flight score, status score, and overall reward used in Wei et al. [4].

We evaluate ENIGMA, BLEU/PPL [184] and Self-Play Evaluation (SPE) based on the correlation between estimated reward and true reward. The results are summarized in Table 5.3. ENIGMA uses the experience data of the other 23 agents to evaluate each agent (i.e., leave-one-bot-out). Note that SPE [4] needs to train a customer agent in addition to the seller agent being evaluated. For a fair comparison, we train the SPE customer agent on both experience data and policy training data (See Appendix D.3 for details). Our empirical

[4]https://github.com/google/airdialogue

[5]We also demonstrate that ENIGMA can be applied to rule based agent in Appendix D.4.2.

Table 5.3: The correlation between two metrics. Each column is a task completion score obtained by interacting human customers ("Selected Agents" denotes only evaluating agents with reasonably good performance).

Setting	Method	Pearson Correlation				Spearman's Rank Correlation			
		Flight Score	Status Score	Reward	Average	Flight Score	Status Score	Reward	Average
All Agents	BLEU	0.1450	-0.1907	-0.0709	-0.0389	0.0370	-0.1453	-0.1472	-0.0852
	PPL	-0.1598	0.1325	0.0195	-0.0026	-0.1817	0.0090	-0.0039	-0.0649
	SPE	0.6450	0.7926	0.7482	0.7286	0.3539	0.8004	0.7400	0.6314
	ENIGMA	**0.9255**	**0.9854**	**0.9672**	**0.9593**	**0.8948**	**0.9839**	**0.9435**	**0.9407**
Selected Agents	BLEU	-0.0621	-0.1442	0.2944	0.0294	-0.1273	-0.2208	0.1793	-0.1758
	PPL	-0.0197	-0.1775	0.0460	-0.0504	-0.1146	-0.4652	-0.0404	-0.2067
	SPE	0.0970	0.5203	0.4777	0.3650	0.1368	0.5304	0.4943	0.3872
	ENIGMA	**0.8640**	**0.9031**	**0.8952**	**0.8874**	**0.8496**	**0.9414**	**0.8782**	**0.8686**

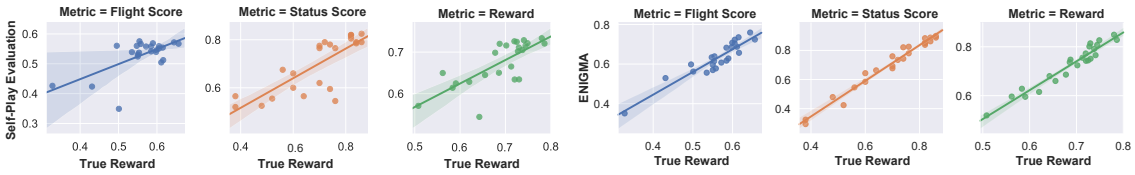
observations are as follows:

• **ENIGMA vs. BLEU/PPL**. ENIGMA significantly outperforms BLEU/PPL. As mentioned earlier, BLEU and PPL are well-known metrics for evaluating language quality. For goal-oriented systems whose goal is to complete a specific task, however, BLEU and PPL scores show little correlation with task completion scores.

• **ENIGMA vs. SPE**. ENIGMA significantly outperforms SPE. To better understand their performance, we also present the regression plots between estimated and true rewards in Figure 5.4. Both ENIGMA and SPE can easily identify agents with extremely poor rewards. However, for selected good agents whose flight score, status score, and overall reward are better than 0.5, 0.7, and 0.65 respectively, SPE performs worse than ENIGMA by a much larger margin (especially for flight score). Additional regression plots are shown in Appendix D.4.1.

• **Ablation Study**. We select 2 out of the 24 agents to illustrate the importance of each component in ENIGMA.

⋆ ***DICE vs. LSTDQ***. Figure 5.5(a) and Figure 5.5(b) show the estimated values of LSTDQ (only fitting the Q-function) and DICE respectively: estimates of LSTDQ are stuck at 0 whereas estimates of DICE approach the true rewards (dotted lines) as training progresses. Figure 5.6 additionally shows that the training objectives of LSTDQ oscillates as DICE stably converges.



(a) SPE vs. Human Evaluation (b) ENIGMA vs. Human Evaluation

Figure 5.4: Regression Plots. The x-axis is the average reward obtained by chatting with human. The y-axis is the reward estimated by SPE / ENIGMA. Different colors denote different types of rewards (flight score, status score, and overall reward). The solid line is obtained by linear regression and the shaded region indicates 95% confidence interval.

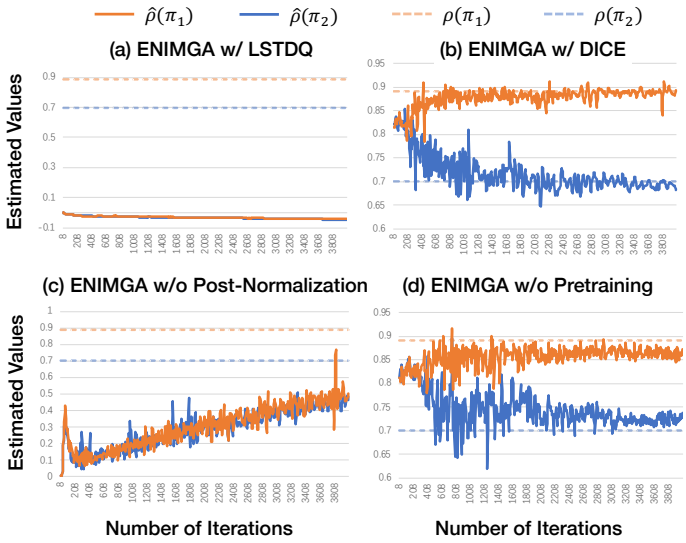


Figure 5.5: Value estimation using different methods for two target agents (π_1 and π_2) vs. # of iterations. Dotted lines denote the true rewards.

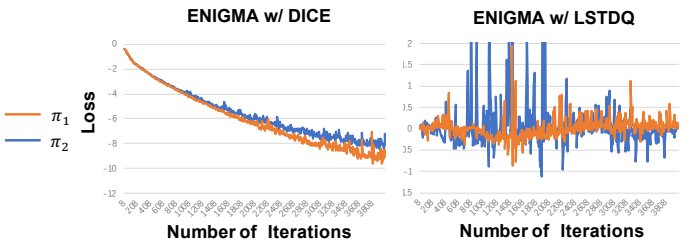
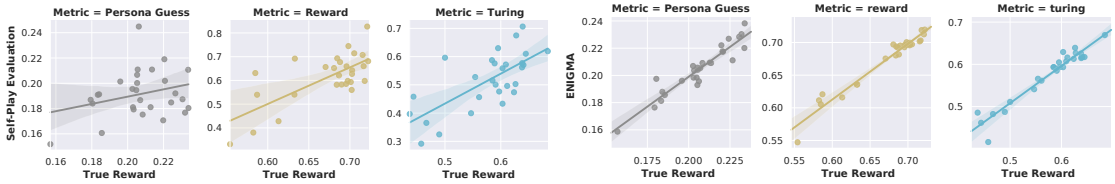


Figure 5.6: Training Objectives vs. Number of Iterations for two target agents.

⋆ ***Post-normalization***. Figure 5.5(c) shows the performance of ENIGMA without post-normalization: The estimated values fail to approach the true rewards.

⋆ ***Pretrained Encoder***. Figure 5.5(d) shows the performance of ENIGMA without the pretrained encoder: The estimated values can approach the true rewards, but are less stable and less accurate than the counterpart with the pretrained encoder.



(a) SPE vs. Human Evaluation (b) ENIGMA vs. Human Evaluation

Figure 5.7: Regression Plots. Only three metrics are presented. Please refer to Appendix D.4.3 for all plots.

5.4.3 Open-Domain Chit-chat Systems

We now test ENIGMA for evaluating open-domain chit-chat dialog systems.

• **Policy Training Data**. We use 29 pre-trained agents[6] provided by See et al. [40]. These agents are trained using behavior cloning on the ***ConvAI2*** dataset[7] [185, 58]. The dataset contains 8,939 pieces of dialog, where participants are instructed to chat naturally using given personas.

• **Experience Data**. We use the experience dataset provided by See et al. [40]. The dataset contains 3,316 agent-human evaluation logs and 10 different language quality metrics for each log.

We follow the setups from Section 4.2 to evaluate ENIGMA, SPE, and 8 Hand-Crafted Dialog Features (HCDFs) based on Pearson and Spearman's rank correlations between the estimated rewards and the true rewards. Here the true rewards are human evaluation scores under 10 different language quality metrics. More details of HCDFs and language quality metrics can be found in See et al. [40]. The average, minimum and maximum of the 10 correlations (under different language quality metrics) of each method are summarized in

[6] https://github.com/facebookresearch/ParlAI/tree/master/projects/controllable_dialogue
[7] https://parl.ai/projects/convai2/

Table 5.4: The correlation between automatic metrics and language score obtained by interacting with human. We only present the average/min/max correlations to all 10 different language metrics in this table. For detailed numbers, please refer to Appendix D.4.3, Figure D.18.

Method	Experience Data	Pearson Correlation			Spearman's Rank Correlation		
		Average	Min	Max	Average	Min	Max
Best of 8 HCDFs	Human-Human	0.6045	0.4468	0.9352	0.3384	0.1724	0.7526
8 HCDFs + Regression	Human-Human	0.5387	-0.0348	0.7519	0.4740	0.2784	0.7880
SPE	Human-Model	0.5907	0.0962	0.8820	0.4350	0.1363	0.6405
SPE	Human-Model (Challenging)	0.3559	-0.1679	0.6900	0.1429	-0.0777	0.3216
ENIGMA	Human-Model	**0.9666**	**0.9415**	**0.9792**	**0.9167**	**0.8717**	**0.9485**
ENIGMA	Human-Model (50% data)	0.9126	0.8506	0.9585	0.7790	0.6651	0.8647
ENIGMA	Human-Model (10% data)	0.7327	0.4544	0.9266	0.5214	0.3651	0.6492
ENIGMA	Human-Model (Challenging)	0.6505	0.5394	0.7762	0.5190	0.3168	0.6672

Table 5.4. Moreover, we also consider using 8 HCDFs to fit the true rewards using linear regression, and the results are also included in Table 5.4.

Note that since we are considering a chit-chat dialog system, SPE does not train an additional agent but asks two identical target agents to chat with each other. However, SPE needs to train an additional model to predict the reward of each dialog. Specifically, we fine-tune the pre-trained RoBERTa encoder with an output layer over the experience data (an additional sigmoid function is applied to ensure an output between 0 and 1). For automatic evaluation of each agent using ENIGMA, we use the experience data of the other 28 agents (i.e., leave-one-bot-out).

• **ENIGMA vs. SPE vs. HCDFs**. ENIGMA significantly outperforms SPE and HCDFs in both Person and Spearman's rank correlations. Moreover, we compare the correlations between estimated rewards and human evaluation scores under each language quality metric. Due to space limit, we only show the plots of ENIGMA and SPE under 3 out of 10 language quality metrics in Figure 5.7. Additional plots and detailed results can be found in Appendix D.4.3. We see that ENIGMA outperforms SPE and HCDFs under all language equality metrics.

• **Sample Efficiency of ENIGMA**. To demonstrate that ENIGMA is sample efficient, we test ENIGMA on randomly sub-sampled (10% and 50%) experience data. We found that

even using only 10% of the experience data, ENIGMA still outperforms SPE and HCDFs.

• **Evaluation under Challenging Experience Data**. To make the evaluation more challenging, we further test ENIGMA by excluding the experience data obtained by the behavior policies similar to the target policy (see more details in Appendix D.4.3). We see that even with such challenging experience data, ENIGMA still outperforms SPE with trained on full data and HCDFs under almost all language quality metrics.

5.5 Discussions

Existing research on automatic evaluation of dialog systems can be categorized into static vs. dynamic evaluation. Most of existing research falls into static evaluation with a focus on language quality of single-turn response or on task-completion given fixed dialog, while few literature emphasizes dynamic properties of an interactive environment. Different from static evaluation, dynamic evaluation considers the sequential interaction between a human and an agent, and thus it is more challenging.

We note that in both static and dynamic evaluations, algorithms rely on the assumption of sufficient data coverage (explicitly or implicitly) to ensure reliable evaluation. For example, in static evaluation, BLEU score requires all reasonably good responses to be exactly covered by the experience data. More recently, Lowe et al. [41] show that their method only works when the behavior policies include the target policy. Dynamic evaluation also assumes the sufficient coverage. We emphasize that it is the information-theoretic limit of all OPE methods [182], which requires the experience data to cover sufficient target policy behaviors to ensure accurate estimation. Therefore, we suggest the broader research community to release human-model interaction evaluation data to further promote research in automatic dialog systems evaluation.

5.6 Conclusion

We develop a model-free OPE framework, ENIGMA, for evaluating dialog systems using experience data. By adopting the current state-of-the-art OPE method in reinforcement learning, we tackle several challenges in modeling dialog systems. Different from existing single-turn language quality metrics and model-based reinforcement learning methods, ENIGMA naturally takes into consideration the interactive and dynamic nature of conversations, while avoiding the difficulty of modeling complex human conversational behaviors. Our thorough experimental results demonstrate that ENIGMA significantly outperforms existing methods in terms of correlation with human evaluation scores. One potential future direction is to extend ENIGMA from off-policy evaluation to off-policy improvement, which aims to learn a dialog system based on experience data [186, 187].

Appendices

APPENDIX A

DATASETS WITH LIMITED SUPERVISION

A.1 GLUE Datasets

Table A.1: Summary of the four benchmarks: GLUE, SNLI, SciTail and ANLI.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
Pairwise Text Classification						
SNLI	NLI	549k	9.8k	9.8k	3	Accuracy
SciTail	NLI	23.5k	1.3k	2.1k	2	Accuracy
ANLI	NLI	163k	3.2k	3.2k	3	Accuracy

The GLUE benchmark, SNLI, SciTail and ANLI is briefly introduced in the following sections. The detailed description can be found in [66, 74, 75, 76]. Table A.1 summarizes the information of these tasks.

• **GLUE**. The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding (NLU) tasks. As shown in Table A.1, it includes question answering [188], linguistic acceptability [189], sentiment analysis [190], text similarity [191], paraphrase detection [192], and natural language inference (NLI) [193, 194, 195, 196, 197, 82]. The diversity of the tasks makes GLUE very suitable for evaluating

the generalization and robustness of NLU models.

• **SNLI**. The Stanford Natural Language Inference (SNLI) dataset contains 570k human annotated sentence pairs, in which the premises are drawn from the captions of the Flickr30 corpus and hypotheses are manually annotated [74]. This is the most widely used entailment dataset for NLI. The dataset is used only for domain adaptation in this study.

• **SciTail** This is a textual entailment dataset derived from a science question answering (SciQ) dataset [75]. The task involves assessing whether a given premise entails a given hypothesis. In contrast to other entailment datasets mentioned previously, the hypotheses in SciTail are created from science questions while the corresponding answer candidates and premises come from relevant web sentences retrieved from a large corpus. As a result, these sentences are linguistically challenging and the lexical similarity of premise and hypothesis is often high, thus making SciTail particularly difficult. The dataset is used only for domain adaptation in this study.

• **ANLI**. The Adversarial Natural Language Inference (ANLI, Nie et al. [76]) is a new large-scale NLI benchmark dataset, collected via an iterative, adversarial human-and-model-in-the-loop procedure. Particular, the data is selected to be difficult to the state-of-the-art models, including BERT and RoBERTa.

APPENDIX B

TECHNICAL DETAILS ABOUT WEAKLY SUPERVISED NER WITH SELF-TRAINING

B.1 Detailed Description of Distant Label Generation

B.1.1 External Knowledge Bases

Wikidata is a collaborative and free knowledge base for the acquisition and maintenance of structured data. It contains over 100 million tokens extracted from the set of verified articles on Wikipedia. Wikidata knowledge imposes a high degree of structured organization. It provides a SPARQL query service for users to obtain entity relationships.

Multi-sources Gazetteers. For each dataset, we build a gazetteer for each entity type. Take CoNLL03 as an example, we build a gazetteer for the type `PER` by collecting data from multiple online sources including Random Name[1], US First Names Database[2], Word Lists[3], US Census Bureau[4], German Surnames[5], Surnames Database[6] and Surname List[7]. We build a gazetteer for the type `ORG` by collecting data from Soccer Team[8], Baseball Team[9] and Intergovernmental Organization[10]. We will release all gazetteers and codes for matching distant labels after the paper is accepted for publication.

[1] https://github.com/dominictarr/random-name
[2] https://data.world/len/us-first-names-database
[3] https://github.com/imsky/wordlists
[4] https://www2.census.gov/topics/genealogy/2010surnames/
[5] https://ziegenfuss.bplaced.net/zfuss/surnames-all.php?tree=1
[6] https://www.surnamedb.com/Surname
[7] https://surnameslist.org/
[8] https://footballdatabase.com/ranking/world
[9] https://www.ducksters.com/sports/list_of_mlb_teams.php
[10] https://en.wikipedia.org/wiki/List_of_intergovernmental_organizations

B.1.2 Distant Labels Generation Details

We first find potential entities by POS tagging obtained from POS tagger, e.g., NLTK [198]. We then match these potential entities by using Wikidata query service. Specifically, we use SPARQL to query the parent categories of an entity in the knowledge tree. We continue querying to the upper levels until a category corresponding to a type is found. For entities with ambiguity (e.g., those linked with multiple parent categories), we discard them during the matching process (i.e., we assign them with type `O`). The above procedure is summarized in Figure 3.2.

We then build, for each entity type in each dataset, a multi-sources gazetteer by crawling online data sources. Following the previous exact string matching methods [103, 101], we match an entity with a type if the entity appears in the gazetteer for that type.

For the unmatched tokens, we further use a set of hand-crafted rules to match entities. We notice that among the true entities, there is usually a stamp word. We match a potential entity with a type if there exists a stamp word in this entity that has frequent occurrence in that type. For example, "Inc." frequently occurs in organization names, thus the appearance of "Inc." indicates that the entity labels of words in the "XXX Inc." should be `B-ORG` or `I-ORG`).

Note that for Twitter, we do not build our own multi-sources gazetteer. We directly use the baseline system proposed in [111] to generate the distant labels.

B.2 Baseline Settings

For the baselines, we implement LSTM-CNN-CRF with Pytorch[11] and use the pre-trained 100 dimension GloVe Embeddings [199] as the input vector. Then, we set the dimension of character-level embeddings to 30 and feed them into a 2D convolutional neural network (CNN) with kernel width as 3. Then, we tune the output dimension in range of $[25, 50, 75, 100, 150]$ and report the best performance. We train the model for 50 epochs

[11] https://pytorch.org/

with early stopping. We use SGD with momentum with $m = 0.9$ and set the learning rate as 2×10^{-3}. We set the dropout rate to 0.5 for linear layers after LSTM. We tune weight decay in range of $[10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]$ and report the best performance.

For other baselines, we follow the officially released implementation from the authors: (1) AutoNER: https://github.com/shangjingbo1226/AutoNER; (2) LRNT: https://github.com/zig-k
win-hu/Low-Resource-Name-Tagging.

B.3 Implementation Details of BOND

All implementation are based on the Huggingface Transformer codebase [12].

B.3.1 Adapting RoBERTa to the NER task

We choose RoBERTa-base as the backbone model of our NER model. A linear classification layer is built upon the pre-trained RoBERTa-base as illustrated in Figure B.1.

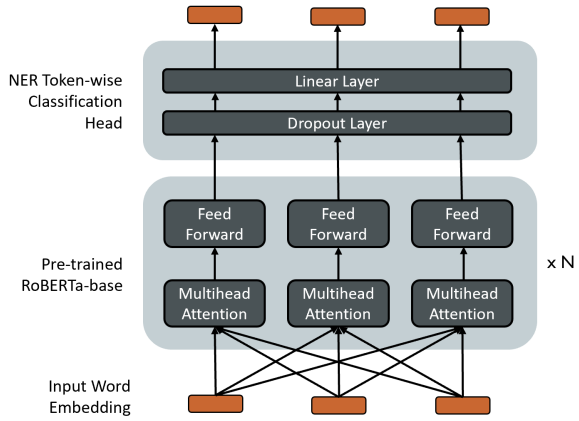


Figure B.1: The NER Model with Pre-trained RoBERTa

[12] https://github.com/huggingface/transformers

B.3.2 Pseudo-labels Generation Details

BERT uses WordPiece [200] for tokenization of the input text. When the teacher model predicts a set of pseudo-labels for all training data in Stage II, it assign labels for padded tokens as well. We ignore those labels in training and loss computation step by label masking.

B.3.3 Parameter Settings

There are several key parameters in our model: 1) For CoNLL03, we choose $T_1 = 900$ (about 1 epoch) and $T_3 = 1756$ (about 2 epochs). For Tweet, we choose $T_1 = 900$ and $T_3 = 900$. For OntoNotes5, we choose $T_1 = 16500$ and $T_3 = 1000$. For Webpage, we choose $T_1 = 300$ and $T_3 = 200$. For Wikigold, we choose $T_1 = 350$ and $T_3 = 700$. As for T_2, we stop training when the number of total training epochs reaches 50 for all datasets. 2) We choose 10^{-5} as the learning rate for CoNLL03, Webpage and Wikigold and 2×10^{-5} for OntoNotes5, Twitter, all with learning rate linear decay of 10^{-4}. 3) We use AdamW with β_1=0.9 and β_2=0.98 as optimizer for all datasets. 4) We set ϵ=0.9 for all datasets. 5) The training batch size is 16 for all datasets except OntoNotes5.0, which uses 32 as the training batch size. 6) For the NER token-wise classification head, we set dropout rate as 0.1 and use a linear classification layer with hidden size 768. For MT, we set ramp-up step as 300 for CoNLL03, 200 for Tweet, 200 for OntoNotes5.0, 300 for Webpage and 40 for Wikigold. We choose the moving average parameters as $\alpha_1 = 0.99$ and $\alpha_2 = 0.995$ for all datasets. For VAT, we set the perturbation size $\epsilon_{vat} = 10^{-4}$.

B.3.4 Multiple Re-initialization

Multiple Re-initialization is implemented as follows: In Stage II, as the performance of the student model no longer improves, we re-initialize it from the pre-trained RoBERTa-base and start a new self-training iteration.

B.3.5 Combine BOND w/ MT&VAT

MT&VAT can easily combined with BOND as follows: During training, we update the student model by minimize the sum of weighted MT (or VAT) loss and Eq. (3.7). The weight of MT (or VAT) loss is selected in $[10, 1, 10^{-1}, 10^{-2}, 10^{-3}]$ using development set.

B.4 Additional Details for NEEDLE

B.4.1 Estimation of Weak Label Confidence

Here we describe how do we estimate the confidence of weak labels — $P(\widetilde{\mathbf{Y}}^c = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}})$. Notice that, the corrected weak labels $\widetilde{\mathbf{Y}}^c$ in NEEDLE consists of two parts: original weak labels $\widetilde{\mathbf{Y}}^w$ and model prediction $\widetilde{\mathbf{Y}}^p$. So we estimate the confidence of corrected weak labels by the confidence of these two parts using a simple linear combination:

$$\begin{aligned} P(\widetilde{\mathbf{Y}}^c = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}) = & \frac{\#\{\text{Matched Tokens}\}}{\#\{\text{Total Tokens}\}} P(\widetilde{\mathbf{Y}}^w = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}) \\ & + (1 - \frac{\#\{\text{Matched Tokens}\}}{\#\{\text{Total Tokens}\}}) P(\widetilde{\mathbf{Y}}^p = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}) \end{aligned}$$

The weight of such linear combination comes from the rule of the weak label completion procedure. Recall that, we use the original weak labels for all matched tokens in original weakly-supervised data, while we use the model prediction for other tokens.

We first assume the confidence of weak labels are very high, i.e. $P(\widetilde{\mathbf{Y}}^w = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}) = 1$, as there is less ambiguity in the domain-specific dictionary and matching process.

The label prediction $\widetilde{\mathbf{Y}}^p$ of CRF model is based on Viterbi decoding score

$$\widetilde{\mathbf{Y}}^p = \arg\max_{\mathbf{Y}} s(\mathbf{Y}) = \text{Decode}(\mathbf{Y}, f(\widetilde{\mathbf{X}}; \theta))$$

The confidence of $\widetilde{\mathbf{Y}}^p$, i.e., $P(\widetilde{\mathbf{Y}}^p = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}})$ can be estimated via histogram binning [122]. Specifically, we categorize samples into bins based on the decoding score $s(\widetilde{\mathbf{Y}}^p)$. For each bin we estimate the confidence using a validation set (independent of the final evaluation set). For a new sample, we first calculate the decoding score, and estimate the prediction confidence by the confidence of the corresponding bin in the histogram. Figure B.2 illustrates an example of histogram binning. As can be seen, the decoding score has a strong correlation with the prediction confidence.

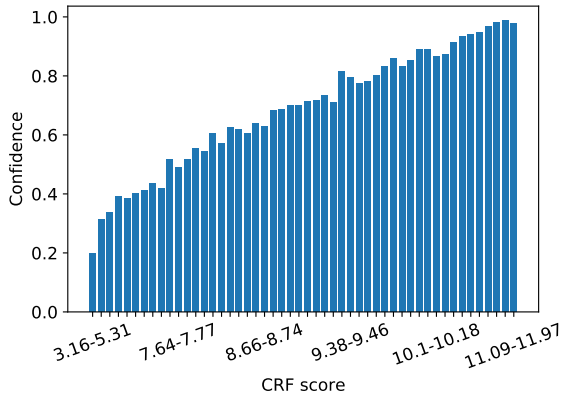


Figure B.2: Decoding Score vs. Accuracy/Confidence

Finally, we enforce a smoothing when estimating the confidence. Specifically, we always make a conservative estimation by a post-processing:

$$P(\widetilde{\mathbf{Y}}^c = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}) = \min(0.95, P(\widetilde{\mathbf{Y}}^c = \widetilde{\mathbf{Y}}|\widetilde{\mathbf{X}}))$$

We enforce such a smoothing to count any potential errors (e.g., inaccurate original weak labels) and prevent model from overfitting. The smoothing parameter is fixed as 0.95 throughout the experiments.

B.4.2 Additional Experimental Results for E-commerce NER

Here we also present Token/Span/Query level Accuracy, as they are commonly used in E-commerce NER tasks.

We also present the additional results for "Performance vs. Size of Strongly Labeled Data" in Table B.2.

Table B.1: Performance of Weighted WSL & Weighted Partial WSL on E-commerce English Query NER

Method	Span P/R/F1	T/S/Q Accu.	
Weighted WSL			
weight = 0.5	75.38/52.94/62.20	61.07/52.61/37.32	
weight = 0.1	77.31/57.85/66.18	65.65/57.70/43.83	
weight = 0.01	78.07/64.41/70.59	71.75/64.43/52.52	
Weighted Partial WSL			
weight = 0.5	72.94/71.77/72.35	81.10/72.53/59.14	
weight = 0.1	75.24/74.68/74.96	83.08/75.36/62.50	
weight = 0.01	76.28/76.34/76.31	84.14/76.94/63.91	

Table B.2: Performance vs. Size of Strongly Labeled Data on E-commerce English Query NER

Method	Span P/R/F1	T/S/Q Accu.
(1%) Query-RoBERTa-CRF (30 epochs)	68.69/70.59/69.63	79.03/71.25/54.36
(10%) Query-RoBERTa-CRF (3 epochs)	71.69/73.72/72.69	81.90/74.26/58.36
(20%) Query-RoBERTa-CRF (3 epochs)	75.16/75.90/75.53	83.65/76.43/62.42
(50%) Query-RoBERTa-CRF (3 epochs)	76.95/77.90/77.42	84.88/78.41/64.96
(1%) NEEDLE	71.20/72.64/71.91	80.74/73.26/57.40
(10%) NEEDLE	76.25/76.15/76.20	84.09/76.67/63.79
(20%) NEEDLE	77.93/77.75/77.84	85.06/78.28/65.88
(50%) NEEDLE	79.12/79.23/79.18	85.92/79.73/67.77

B.4.3 Additional Experimental Results for Biomedical NER

Table B.3: Main Results on Biomedical NER: Span Precision/Recall/F1. The *Best* performance is **bold**, and the results that are close to best performance ($\leq 0.2\%$) are also **bold**.

Method	BC5CDR-chem	BC5CDR-disease	NCBI-disease
Reported F1-scores of Baselines [128]. Previous SOTA: PubMedBERT/BioBERT.			
BERT	-/-/89.99	-/-/79.92	-/-/85.87
BioBERT	-/-/92.85	-/-/84.70	-/-/89.13
SciBERT	-/-/92.51	-/-/84.70	-/-/88.25
PubMedBERT	-/-/93.33	-/-/85.62	-/-/87.82
Re-implemented Baselines			
BERT	88.55/90.49/89.51	77.54/81.87/79.64	83.50/88.54/85.94
BERT-CRF	88.59/91.44/89.99	78.70/81.53/80.09	85.33/86.67/85.99
BioBERT	92.59/93.11/92.85	82.36/86.66/84.45	86.75/90.83/88.74
BioBERT-CRF	92.64/93.28/92.96	83.73/86.80/85.23	87.18/91.35/89.22
Based on BioBERT and CRF layer			
ST	92.40/93.74/93.06	84.01/87.18/85.56	87.00/91.98/89.42
WSL	82.17/88.91/85.41	90.72/87.27/88.96	87.14/71.98/78.84
NEEDLE w/o WLC/NAL	**92.85**/93.31/93.08	91.37/88.34/89.83	**91.68**/91.77/91.73
NEEDLE w/o FT/NAL	79.29/84.38/81.75	82.44/**94.03**/87.85	87.17/90.62/88.86
NEEDLE w/o NAL	**92.93**/94.28/**93.60**	86.73/93.69/90.07	**91.82**/92.40/**92.11**
NEEDLE w/o FT	79.87/84.31/82.03	82.39/**94.12**/87.86	87.31/91.04/89.14
NEEDLE	**92.89/94.60/93.74**	**87.99**/93.56/**90.69**	**91.76/92.81/92.28**

B.4.4 Extension: Multilingual NER

The proposed framework can be extended to improve multilingual NER. For Stage I and Stage II, we use data from other languages to learn domain-specific knowledge and task-related knowledge. In the final fine-tuning stage, we use the data from the target language, which allows us to adapt the model to the target language and obtain a better performance on the target language. The framework is summarized in Figure B.3. The results of Multilingual Query NER are presented in Table B.4. As can be seen, NEEDLE outperforms baseline methods.

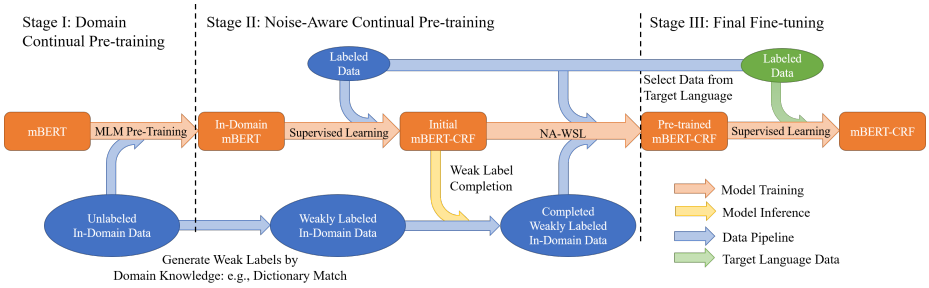


Figure B.3: Three-Stage NEEDLE for Multilingual NER

Table B.4: E-commerce Multilingual Query NER: Span Precision/Recall/F1 and Token/Span/Query level Accuracy. The *Best* performance is **bold**, and the results that are close to best performance ($\leq 0.2\%$) are also **bold**. 'mBERT-CRF (Single)': fine-tune mBERT with strongly labeled data from the target language. 'w/ Fine-tune': the additional fine-tuning stage only use strongly labeled data from the target language. For other methods, we use multilingual human-annotated data.

Method (*Span P/R/F1*)	En	Fr	It	De	Es
mBERT-CRF (Single)	76.14/76.04/76.09	72.87/73.00/72.93	76.95/77.67/77.31	74.74/78.08/76.37	76.34/76.75/76.54
mBERT-CRF	76.38/76.25/76.31	74.69/75.06/74.87	77.82/77.60/77.71	75.93/78.52/77.20	78.18/77.57/77.87
Query-mBERT-CRF	77.21/77.18/77.19	74.59/75.05/74.82	78.22/78.01/78.11	76.46/79.12/77.77	78.50/77.73/78.11
Based on Query-mBERT and CRF layer					
ST	77.52/77.33/77.42	75.15/75.28/75.21	78.00/77.64/77.82	76.82/79.43/78.10	79.14/78.17/78.65
WSL	74.20/48.09/58.35	71.17/51.71/59.90	74.72/51.51/60.98	74.34/52.68/61.66	76.32/53.85/63.14
NEEDLE w/o WLC/NAL	77.89/77.47/77.68	75.28/75.35/75.31	78.17/78.28/78.22	76.68/79.33/77.99	78.29/78.14/78.22
NEEDLE w/o FT/NAL	72.73/75.06/73.87	72.00/73.12/72.56	75.19/75.34/75.26	74.65/77.63/76.11	77.07/76.18/76.62
NEEDLE w/o NAL	**78.27/77.74/78.00**	**76.09/75.95/76.02**	79.14/79.25/79.19	77.55/79.63/78.58	79.60/78.86/79.23
NEEDLE w/o FT	72.79/75.01/73.88	72.46/73.46/72.96	75.39/75.50/75.44	75.09/77.98/76.51	77.46/76.29/76.87
NEEDLE	**78.40/77.95/78.17**	**76.05/75.91/75.98**	**79.61/79.76/79.68**	**77.79/79.90/78.83**	**79.85/79.13/79.49**
Method (*T/S/Q Accu.*)	En	Fr	It	De	Es
mBERT-CRF (Single)	83.26/76.80/61.68	80.27/72.91/57.48	83.70/78.13/60.75	79.53/76.38/60.72	83.58/77.56/59.64
mBERT-CRF	83.37/76.97/62.21	81.43/74.92/60.35	84.31/78.06/60.65	80.48/76.82/62.47	84.94/78.23/61.44
Query-mBERT-CRF	84.15/77.85/63.44	81.36/74.91/60.17	84.83/78.46/61.26	80.93/77.40/62.81	85.20/78.27/62.12
Based on Query-mBERT and CRF layer					
ST	84.18/78.02/63.57	81.66/75.12/60.92	84.45/78.13/60.89	81.26/77.72/63.61	85.35/78.56/62.90
WSL	54.40/47.43/28.97	59.11/51.08/32.85	59.79/50.59/30.75	56.16/51.16/33.59	61.36/53.29/32.48
NEEDLE w/o WLC/NAL	84.42/78.12/64.43	81.65/75.24/60.74	84.76/78.65/61.77	81.32/77.59/63.37	84.82/78.84/61.95
NEEDLE w/o NAL/FT	83.46/75.80/57.93	81.20/73.04/56.90	83.48/75.97/57.22	80.31/76.00/60.79	83.90/76.80/59.30
NEEDLE w/o NAL	**84.63/78.42/64.76**	**82.34/75.83/61.91**	85.34/79.63/63.17	81.68/77.90/64.34	85.64/79.48/63.41
NEEDLE w/o FT	83.50/75.76/58.01	80.92/73.38/57.34	83.45/76.03/57.39	80.48/76.31/61.22	84.10/76.97/60.12
NEEDLE	**84.74/78.59/64.86**	**82.14/75.80/61.96**	**85.65/80.12/63.71**	**81.79/78.15/64.84**	**86.00/79.80/64.03**

APPENDIX C

TECHNICAL DETAILS ABOUT WEAKLY SUPERVISED LEARNING WITH CONTRASTIVE REGULARIZED SELF-TRAINING

C.1 Datasets and Weak Supervision Details

C.1.1 Data Source

The seven benchmarks in our experiments are all publicly available. Below are the links to downloadable versions of these datasets.

◇ *AGNews* and *Yelp*: We use the datasets from Meng et al. [138]: https://github.com/yumeng5/WeSTClass.

◇ *IMDB*: Dataset is available at https://ai.stanford.edu/~amaas/data/sentiment/.

◇ *TREC*: Dataset is available at https://cogcomp.seas.upenn.edu/Data/QA/QC/.

◇ *MIT-R*: Dataset is available at http://groups.csail.mit.edu/sls/downloads/restaurant/.

◇ *ChemProt*: The raw dataset is available at http://www.cbs.dtu.dk/services/ChemProt/ChemProt-2.0/. The preprocessed dataset is available at https://drive.google.com/drive/folders/1Uzi76WE5oOe7Rv_vhXs7W4LKQBt6udFX?usp=sharing.

◇ *WiC*: Dataset is available at https://pilehvar.github.io/wic/.

C.1.2 Train/Test Split

For *AGNews*, *Yelp*, *IMDB* and *ChemProt*, we follow the split ratio in Meng et al. [138]. We use 80% of the data as the training set, 10% as the validation set, and 10% as the test set.

For *MIT-R* and *TREC*, we split the data in the same way as Awasthi et al. [24].

For *WiC*, we use the same dataset and the train/test split ratio in Pilehvar and Camacho-Collados [152].

C.1.3 Weak Supervision Details

COSINE does not require any human annotated examples in the training process, and it only needs weak supervision sources such as keywords and semantic rules. According to some studies in existing works Awasthi et al. [24] and Zhou et al. [201], such weak supervisions are cheap to obtain and are much efficient than collecting clean labels. In this way, we can obtain significantly more labeled examples using these weak supervision sources than human labor.

There are two types of semantic rules that we apply as weak supervisions:

- ◇ *Keyword Rule*: `HAS(x, L)` $\rightarrow$ `C`. If x matches one of the words in the list L, we label it as C.
- ◇ *Pattern Rule*: `MATCH(x, R)` $\rightarrow$ `C`. If x matches the regular expression R, we label it as C.

In addition to the keyword rule and the pattern rule, we can also use third-party tools to obtain weak labels. These tools (e.g., TextBlob[1]) are available online and can be obtained cheaply, but their prediction is not accurate enough (when directly use this tool to predict label for all training samples, the accuracy on Yelp dataset is around 60%).

We now introduce the semantic rules on each dataset:

- ◇ *AGNews*: Examples are demonstrated in Table C.1.
- ◇ *IMDB*: Examples are demonstrated in Table C.2.
- ◇ *Yelp*: Examples are demonstrated in Table C.3. We provide labeling rules in eight views.
- ◇ *MIT-R*: There are 15 rules in Awasthi et al. [24]. Please refer to the original paper for detailed information on rules.

[1] https://textblob.readthedocs.io/en/dev/index.html.

◇ *TREC*: There are 68 rules in Awasthi et al. [24]. Please refer to the original paper for detailed information on rules.

◇ *ChemProt*: There are 26 rules. We show part of the rules in Table C.4.

◇ *WiC*: Each sense of each word in WordNet has example sentences. For each sentence in the WiC dataset and its corresponding keyword, we collect the example sentences of that word from WordNet. Then for a pair of sentences, the corresponding weak label is "True" if their definitions are the same, otherwise the weak label is "False".

Table C.1: Examples of semantic rules on AGNews.

Rule
`[war, prime minister, president, commander, minister, military, militant, kill, operator]` → `POLITICS`
`[baseball, basketball, soccer, football, boxing, swimming, world cup, nba,olympics,final, fifa]` → `SPORTS`
`[delta, cola, toyota, costco, gucci, citibank, airlines]` → `BUSINESS`
`[technology, engineering, science, research, cpu, windows, unix, system, computing, compute]` → `TECHNOLOGY`

Table C.2: Examples of semantic rules on IMDB.

Rule
[masterpiece, outstanding, perfect, great, good, nice, best, excellent, worthy, awesome, enjoy, positive, pleasant, wonderful, amazing, superb, fantastic, marvellous, fabulous] → POS
[bad, worst, horrible, awful, terrible, crap, shit, garbage, rubbish, waste] → NEG
[beautiful, handsome, talented]→ POS
[fast forward, n t finish] → NEG
[well written, absorbing,attractive, innovative, instructive,interesting, touching, moving]→ POS
[to sleep, fell asleep, boring, dull, plain]→ NEG
[than this, than the film, than the movie]→ NEG
MATCH(x, *PRE*EXP*) → POS PRE = [will , ll , would , d , can t wait to] EXP = [next time, again, rewatch, anymore, rewind]
PRE = [highly , do , would , definitely , certainly , strongly , i , we] EXP = [recommend, nominate]
PRE = [high , timeless , priceless , has , great , of , real , instructive] EXP = [value, quality, meaning, significance]

Table C.3: Examples of semantic rules on Yelp.

View	**Rule**
General	`[outstanding, perfect, great, good, nice, best, excellent, worthy, awesome, enjoy, positive, pleasant,wonderful, amazing, recommend]` $\rightarrow$ `POS`
General	`[bad, worst, horrible, awful, terrible, nasty, shit, distasteful,dreadful, negative]`$\rightarrow$ `NEG`
Mood	`[happy, pleased, delighted,contented, glad, thankful, satisfied]` $\rightarrow$ `POS`
Mood	`[sad, annoy, disappointed,frustrated, upset, irritated, harassed, angry, pissed]`$\rightarrow$ `NEG`
Service	`[friendly, patient, considerate, enthusiastic, attentive, thoughtful, kind, caring, helpful, polite, efficient, prompt]` $\rightarrow$ `POS`
Service	`[slow, offended, rude, indifferent, arrogant]`$\rightarrow$ `NEG`
Price	`[cheap, reasonable, inexpensive, economical]` $\rightarrow$ `POS`
Price	`[overpriced, expensive, costly, high-priced]`$\rightarrow$ `NEG`
Environment	`[clean, neat, quiet, comfortable, convenien, tidy, orderly, cosy, homely]` $\rightarrow$ `POS`
Environment	`[noisy, mess, chaos, dirty, foul]`$\rightarrow$ `NEG`
Food	`[tasty, yummy, delicious,appetizing, good-tasting, delectable, savoury, luscious, palatable]` $\rightarrow$ `POS`
Food	`[disgusting, gross, insipid]`$\rightarrow$ `NEG`
We use TextBlob as the third-party tool to obtan the sentiment subjectivity score	POLARITY$(x) > 0.5 \rightarrow$ POS, POLARITY$(x) < -0.5 \rightarrow$ NEG

Table C.4: Examples of semantic rules on Chemprot.

Rule	Example
`HAS (x, [amino acid,mutant, mutat, replace] ) → part_of`	A major part of this processing requires endoproteolytic cleavage at specific pairs of basic [CHEMICAL]amino acid[CHEMICAL] residues, an event necessary for the maturation of a variety of important biologically active proteins, such as insulin and [GENE]nerve growth factor[GENE].
`HAS (x, [bind, interact, affinit] ) → regulator`	The interaction of [CHEMICAL]naloxone estrone azine[CHEMICAL] (N-EH) with various [GENE]opioid receptor[GENE] types was studied in vitro.
`HAS (x, [activat, increas, induc, stimulat, upregulat] ) → upregulator/activator`	The results of this study suggest that [CHEMICAL]noradrenaline[CHEMICAL] predominantly, but not exclusively, mediates contraction of rat aorta through the activation of an [GENE]alphalD-adrenoceptor[GENE].
`HAS (x, [downregulat, inhibit, reduc, decreas] ) → downregulator/inhibitor`	These results suggest that [CHEMICAL]prostacyclin[CHEMICAL] may play a role in downregulating [GENE]tissue factor[GENE] expression in monocytes, at least in part via elevation of intracellular levels of cyclic AMP.
`HAS (x, [ agoni, tagoni]* ) → agonist *` (note the leading whitespace in both cases)	Alprenolol and BAAM also caused surmountable antagonism of [CHEMICAL]isoprenaline[CHEMICAL] responses, and this [GENE]beta 1-adrenoceptor[GENE] antagonism was slowly reversible.
`HAS (x, [antagon] ) → antagonist`	It is concluded that [CHEMICAL]labetalol[CHEMICAL] and dilevalol are [GENE]beta 1-adrenoceptor[GENE] selective antagonists.
`HAS (x, [modulat, allosteric] ) → modulator`	[CHEMICAL]Hydrogen sulfide[CHEMICAL] as an allosteric modulator of [GENE]ATP-sensitive potassium channels[GENE] in colonic inflammation.
`HAS (x, [cofactor] ) → cofactor`	The activation appears to be due to an increase of [GENE]GAD[GENE] affinity for its cofactor, [CHEMICAL]pyridoxal phosphate[CHEMICAL] (PLP).
`HAS (x, [substrate, catalyz, transport, produc, conver] ) → substrate/product`	Kinetic constants of the mutant [GENE]CrAT[GENE] showed modification in favor of longer [CHEMICAL]acyl-CoAs[CHEMICAL] as substrates.
`HAS (x, [not] ) → not`	[CHEMICAL]Nicotine[CHEMICAL] does not account for the CSE stimulation of [GENE]VEGF[GENE] in HFL-1.

C.2 Baseline Settings

We implement Self-ensemble, FreeLB, Mixup and UST based on their original paper. For other baselines, we use their official release:

◇ WeSTClass [138]: https://github.com/yumeng5/WeSTClass.

◇ RoBERTa [16]: https://github.com/huggingface/transformers.

◇ SMART [27]: https://github.com/namisan/mt-dnn.

◇ Snorkel [28]: https://www.snorkel.org/.

◇ ImplyLoss [24]: https://github.com/awasthiabhijeet/Learning-From-Rules[2]. ◇ Denoise [34]: https://github.com/weakrules/Denoise-multi-weak-sources.

C.3 Adapting RoBERTa to Different Tasks

C.3.1 Data Tokenization

For different tasks we adopt different tokenization strategies as follows:

- ◇ For sentimental analysis, topic classification, question classification, and slot filling, we add a `[CLS]` token at the beginning of every sequences.
- ◇ For relation classification, we need to identify the relation between two given terms. We add the `[CLS]` tokens, as well as special tokens `[ENT1]` and `[ENT2]` around each of the terms.
- ◇ For word sense disambiguation, the input data consist of two sentences, and each of them contains a target word. We add the `[CLS]` tokens, as well as a special token `[ENT]` before each of the term. We also add a `[SEP]` token between the two sentences.

[2]We use the RoBERTa embedding instead of the ELMo embedding for fair comparison.

C.3.2 Classification Heads

To adapt RoBERTa to downstream tasks, we use different classification heads as follows:

- ◇ For sentimental analysis, topic classification, and question classificaiton, we use a sequence classification head and we use the embedding of the `[CLS]` token as the representation of the input sequence.
- ◇ For slot filling, we use a token classification head. During loss computation and evaluation, we ignore the outputs corresponding to the special tokens.
- ◇ For relation classification and word sense disambiguation, we use the same classification head as Wu and He [202]. See https://github.com/monologg/R-BERT for details.

C.4 Detailed Information on Experiment Setups

C.4.1 Computing Infrastructure

System: Ubuntu 18.04.3 LTS; Python 3.7; Pytorch 1.2.
CPU: Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz.
GPU: GeForce GTX TITAN X.

C.4.2 Hyper-parameters

We use AdamW [203] as the optimizer, and the learning rate is chosen from $1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}\}$. A linear learning rate decay schedule with warm-up 0.1 is used, and the number of training epochs is 5.

Hyper-parameters are shown in Table C.5. We use a grid search to find the optimal setting for each task. Specifically, we search T_1 from 10 to 2000, T_2 from 1000 to 5000, T_3

from 10 to 500, ξ from 0 to 1, and λ from 0 to 0.5. All results are reported as the average over three runs.

Table C.5: Hyper-parameter configurations. Note that we only keep certain number of tokens.

Hyper-parameter	**AGNews**	**IMDB**	**Yelp**	**MIT-R**	**TREC**	**Chemprot**	**WiC**
Dropout Ratio	0.1						
Maximum Tokens	128	256	512	64	64	400	256
Batch Size	32	16	16	64	16	24	32
Weight Decay	10^{-4}						
Learning Rate	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
T_1	160	160	200	150	500	400	1700
T_2	3000	2000	2000	1500	2500	1000	3000
T_3	250	50	100	15	30	15	80
ξ	0.6	0.7	0.7	0.2	0.3	0.7	0.7
λ	0.1	0.05	0.05	0.1	0.05	0.05	0.05

C.4.3 Number of Parameters

COSINE and most of the baselines (RoBERTa-WL / RoBERTa-CL / SMART / WeSTClass / Self-Ensemble / FreeLB / Mixup / UST) are built on the RoBERTa-base model with about 125M parameters. Snorkel is a generative model with about 1M parameters. ImplyLoss and Denoise freezes the embedding and has less than 1M parameters. However, these models cannot achieve satisfactory performance in our experiments.

C.4.4 Runtime Analysis

To accelerate contrastive learning, we adopt the doubly stochastic sampling approximation to reduce the computational cost. Specifically, the high confidence samples $\mathcal{C}$ in each batch $\mathcal{B}$ yield $\mathcal{O}(|\mathcal{C}|^2)$ sample pairs, and we sample $|\mathcal{C}|$ pairs from them. Table C.6, C.7 summarizes the running time of our model and the baselines. We can see that our framework does not impose much additional time costs. From the result, our COSINE framework not

require much extra time cost compared with compared with baselines.

Table C.6: Running time of COSINE. †: Evaluation time is included, the actual training time is much shorter.

Time	AGNews	IMDB	Yelp	MIT-R	TREC	Chemprot	WiC
Running Time per Iteration (s)	0.68	0.90	0.52	0.51	0.81	0.80	0.65
Total Running Time (h)†	1.54	1.02	0.91	0.35	0.45	0.71	0.67

Table C.7: Runtime of baselines (in hours).

Method	AGNews	IMDB	Yelp	MIT-R	TREC	Chemprot	WiC
RoBERTa-WL/FL	0.92	0.57	0.60	0.93	0.38	0.52	0.44
FreeLB	1.40	1.21	1.05	1.65	0.73	0.85	0.79
Self-ensemble	1.03	0.93	0.87	1.12	0.61	0.71	0.63
Mixup	0.97	0.74	0.71	1.04	0.48	0.59	0.53
SMART	1.61	1.29	1.10	2.20	0.82	0.98	0.90
Snorkel	0.09	0.23	0.34	0.001	0.05	0.003	—
WESTClass	1.44	0.91	0.88	—	0.40	—	0.57
ImplyLoss	0.43	0.07	0.16	0.15	0.68	0.12	0.03
Denoise	0.20	0.06	0.07	0.03	0.04	0.06	0.05
UST	1.66	1.31	1.28	0.45	0.70	0.86	0.79

C.4.5 Performance on the Development Set

We summarize model performance on the development set in Table C.8.

Table C.8: Performance of COSINE on the development set.

Result	**AGNews**	**IMDB**	**Yelp**	**MIT-R**	**TREC**	**Chemprot**	**WiC**
Dev	87.71	89.96	95.29	75.90	70.98	53.52	67.71
Test	87.52	90.54	95.97	76.63	82.59	54.36	64.50

C.5 Early Stopping and Earlier Stopping

Our model adopts the earlier stopping strategy during the initialization stage. Here we use "earlier stopping" to differentiate from "early stopping", which is standard in fine-tuning algorithms. Early stopping refers to the technique where we stop training when the evaluation score drops. Earlier stopping is self-explanatory, namely we fine-tune the pre-trained LMs with only a few steps, even before the evaluation score starts dropping. This technique can efficiently prevent the model from overfitting. For example, as Figure 4.8b illustrates, on the IMDB dataset, our model overfits after 240 iterations of initialization when using weak labels. In contrast, the model achieves good performance even after 400 iterations of fine-tuning when using clean labels. This verifies the necessity of earlier stopping.

C.6 Comparison of Distance Measures in Contrastive Learning

The contrastive regularizer $\mathcal{R}_1(\theta; \widetilde{\boldsymbol{y}})$ is related to two designs: the sample distance metric d_{ij} and the sample similarity measure W_{ij}. In our implementation, we use the scaled Euclidean distance as the default for d_{ij} and Eq. 4.5 as the default for W_{ij}. Here we discuss other designs.

C.6.1 Sample distance metric d

Given the encoded vectorized representations $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ for samples i and j, we consider two distance metrics as follows.

Scaled Euclidean distance (Euclidean): We calculate the distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ as

$$d_{ij} = \frac{1}{d} \left\| \boldsymbol{v}_i - \boldsymbol{v}_j \right\|_2^2 . \tag{C.1}$$

Cosine distance (Cos)[3]: Besides the scaled Euclidean distance, cosine distance is another widely-used distance metric:

$$d_{ij} = 1 - \cos\left(\boldsymbol{v}_i, \boldsymbol{v}_j\right) = 1 - \frac{\left\| \boldsymbol{v}_i \cdot \boldsymbol{v}_j \right\|}{\|\boldsymbol{v}_i\| \|\boldsymbol{v}_j\|}. \tag{C.2}$$

C.6.2 Sample similarity measures W

Given the soft pseudo-labels $\widetilde{\boldsymbol{y}}_i$ and $\widetilde{\boldsymbol{y}}_j$ for samples i and j, the following are some designs for W_{ij}. In all of the cases, W_{ij} is scaled into range $[0, 1]$ (we set $\gamma = 1$ in Eq. 4.7 for the hard similarity).

Hard Similarity: The hard similarity between two samples is calculated as

$$W_{ij} = \begin{cases} 1, & \text{if } \underset{k \in \mathcal{Y}}{\text{argmax}} [\widetilde{\boldsymbol{y}}_i]_k = \underset{k \in \mathcal{Y}}{\text{argmax}} [\widetilde{\boldsymbol{y}}_j]_k, \\ 0, & \text{otherwise.} \end{cases} \tag{C.3}$$

This is called a "hard" similarity because we obtain a binary label, i.e., we say two samples are similar if their corresponding hard pseudo-labels are the same, otherwise we say they are dissimilar.

[3] We use Cos to distinguish from our model name COSINE.

Soft KL-based Similarity: We calculate the similarity based on KL distance as follows.

$$W_{ij} = \exp\left(-\frac{\beta}{2}\Big(\mathcal{D}_{KL}(\widetilde{\boldsymbol{y}}_i \| \widetilde{\boldsymbol{y}}_j) + \mathcal{D}_{KL}(\widetilde{\boldsymbol{y}}_j \| \widetilde{\boldsymbol{y}}_i)\Big)\right), \tag{C.4}$$

where β is a scaling factor, and we set $\beta = 10$ by default.

Soft L2-based Similarity: We calculate the similarity based on L2 distance as follows.

$$W_{ij} = 1 - \frac{1}{2}||\widetilde{\boldsymbol{y}}_i - \widetilde{\boldsymbol{y}}_j||_2^2, \tag{C.5}$$

C.6.3 COSINE under different d and W.

We show the performance of COSINE with different choices of d and W on Agnews and MIT-R in Table C.9. We can see that COSINE is robust to these choices. In our experiments, we use the scaled euclidean distance and the hard similarity by default.

Table C.9: Performance of COSINE under different settings.

Distance d	Euclidean			Cos		
Similarity W	Hard	KL-based	L2-based	Hard	KL-based	L2-based
AGNews	**87.52**	86.44	86.72	87.34	86.98	86.55
MIT-R	76.61	76.68	76.49	76.55	**76.76**	76.58

C.7 Significance Test of COSINE

Table C.10 shows the independent two sample T-test on the result of the best baselines and our method in 7 datasets. All the P-values are less than significance level $\alpha = 0.01$ except Chemprot, where the P-values (0.03) is also less than $\alpha = 0.05$. From the result, it indicates that the improvements of COSINE compared with baselines are significant.

Table C.10: Statistical significance test of COSINE on different datasets. ** (*) means the result is significant according to the T-test at level 0.01 (0.05) compared to the best baseline.

Datasets	**Our Result**	**Comparison Method**	T**-statistic**	P**-value**
AGNews	87.52 ± 0.24**	COSINE v.s. UST [154]	5.5904	0.0050
IMDB	90.54 ± 0.38**	COSINE v.s. FreeLB [26]	10.1240	0.0005
Yelp	95.97 ± 0.32**	COSINE v.s. Mixup [153]	12.8588	0.0002
MIT-R	76.61 ± 0.58**	COSINE v.s. UST [154]	5.0964	0.0070
TREC	82.59 ± 0.71**	COSINE v.s. Implyloss [24]	4.7670	0.0089
Chemprot	54.36 ± 0.35*	COSINE v.s. Implyloss [24]	3.1686	0.0339
WIC	67.71 ± 0.55**	COSINE v.s. Mixup [153]	7.5189	0.0017

APPENDIX D

TECHNICAL DETAILS ABOUT AUTOMATIC DIALOGUE EVALUATION WITH OFF-POLICY EVALUATION

D.1 ENIGMA

D.1.1 Supporting Theorem

Theorem 1. The augmented MDP with infinite horizon satisfies the following properties:

[topsep=0pt,parsep=2pt,partopsep=0pt, leftmargin=*]

- It has a unique stationary state-action visitation distribution $d^\pi(s, a)$;
- For the station-action pair (s_t, a_t) in a conversation h with padded pseudo states, we have

$$d^\pi(s_t, a_t) = \frac{1}{T_{\max}} \sum_{\{(s_k,a_k)\}_{k=1}^{t-1}} [\mu_0(s_1)\pi(a_1|s_1)P(s_2|a_1, s_1) \cdots P(s_t|a_{t-1}, s_{t-1})\pi(a_t|s_t)], \tag{D.1}$$

  where $\{(s_k, a_k)\}_{k=1}^{t-1}$ are the state-action pairs in the same conversation as (s_t, a_t);
- The policy value can be computed by sampling from $d^\pi(s, a)$, and we have

$$\rho_A(\pi) = \mathbb{E}_{(s,a)\sim d^\pi(s,a)}[R(s, a)] = \rho(\pi)/T_{\max}. \tag{D.2}$$

Proof. **First**, we prove that the augmented MDP has a unique stationary state-action visitation distribution shown in (D.1).

As the augmented MDP is periodic with period $T_{\max}$, the uniqueness and stationary distribution can not be immediately obtained by ergodicity of the MDP (the first two points

of the Theorem).

To obtain the stationary state-action visitation distribution, we essentially need to solve the following equations:

$$d^{\pi}(s,a) = \sum_{(s',a')} d^{\pi}(s',a')P(s|s',a')\pi(a|s), \text{ for all } (s,a) \tag{D.3}$$

with $d^{\pi}(s,a)$ is a probability measure on the state-action space, i.e., $\sum_{(s,a)} d^{\pi}(s,a) = 1$.

We first group the state-action pairs by their dialog turns t. More specifically, we define $\mathcal{S}_t := \{s_t : s_t \text{ contains } t \text{ dialog turns}\}$, $\mathcal{A}_t := \{a_t : a_t \text{ is the response at the } t\text{−th dialog turn}\}$ and $\mathcal{Q}_t = \mathcal{S}_t \times \mathcal{A}_t$. We have the state space is the direct sum of state groups $\mathcal{S}_0 \oplus \mathcal{S}_1 \cdots \oplus \mathcal{S}_{T_{\max}} = \mathcal{S}$ and the action space is the union of all action groups $\bigcup_{t=1}^{T_{\max}} \mathcal{A}_t = \mathcal{A}$. We further have $\mathcal{Q}_0 \oplus \mathcal{Q}_1 \cdots \oplus \mathcal{Q}_{T_{\max}} = \mathcal{S} \times \mathcal{A} = \mathcal{Q}$. Notice that t is the number of dialog turns in original MDP, not the time step for the augmented MDP.

We then consider the quantity $S_t = \sum_{(s_t,a_t)\in\mathcal{Q}_t} d^{\pi}(s_t,a_t)$, which sum over the LHS of the Eq.(D.3) for each group of (s_t,a_t). We now expand the corresponding sum of the RHS of (D.3):

$$\begin{aligned}
S_t &= \sum_{(s_t,a_t)\in\mathcal{Q}_t} d^{\pi}(s_t,a_t) \\
&= \sum_{(s_t,a_t)\in\mathcal{Q}_t} \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} d^{\pi}(s_{t-1},a_{t-1})P(s_t|s_{t-1},a_{t-1})\pi(a_t|s_t) \\
&= \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} \sum_{(s_t,a_t)\in\mathcal{Q}_t} d^{\pi}(s_{t-1},a_{t-1})P(s_t|s_{t-1},a_{t-1})\pi(a_t|s_t) \\
&= \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} d^{\pi}(s_{t-1},a_{t-1}) \\
&= S_{t-1} \ \ (t>1).
\end{aligned}$$

We have $S_1 = S_2 = \cdots = S_{T_{\max}} = S$. As $d^{\pi}(s,a)$ is a probability measure, we have the

following unique solution for S_t's

$$S = \frac{1}{T_{\max}} \sum_{t=1}^{T_{\max}} S_t = \frac{1}{T_{\max}} \sum_{t=1}^{T_{\max}} \sum_{(s_t,a_t)\in\mathcal{Q}_t} d^\pi(s_t, a_t) = \frac{1}{T_{\max}} \sum_{(s,a)\in\mathcal{Q}} d^\pi(s, a) = \frac{1}{T_{\max}}.$$

As there is only one possibility for the last state-action pairs in all possible conversation that $s_{T_{\max}} = \mathrm{Pad}_{T_{\max}}, a_{T_{\max}} = \mathrm{NextPad}$, we have $d^\pi(\mathrm{Pad}_{T_{\max}}, \mathrm{NextPad}) = S_{T_{\max}} = \frac{1}{T_{\max}}$. We now consider (s_1, a_1), which is the first state-action pair of a conversation. We have

$$\begin{aligned} d^\pi(s_1, a_1) &= \sum_{(s',a')} d^\pi(s', a') P(s_1|s', a')\pi(a_1|s_1) \\ &= d^\pi(\mathrm{Pad}_{T_{\max}}, \mathrm{NextPad}) P(s_1|\mathrm{Pad}_{T_{\max}}, \mathrm{NextPad})\pi(a_1|s_1) = \frac{1}{T_{\max}}\mu_0(s_1)\pi(a_1|s_1), \end{aligned} \tag{D.4}$$

which is the unique solution. For any $(s_t, a_t) \in \mathcal{Q}_t (t > 1)$, the previous state-action pairs in the same conversation must be in $\mathcal{Q}_{t-1}$. We have

$$\begin{aligned} d^\pi(s_t, a_t) &= \sum_{(s',a')} d^\pi(s', a') P(s_t|s', a')\pi(a_t|s_t) \\ &= \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} d^\pi(s_{t-1}, a_{t-1}) P(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t). \end{aligned} \tag{D.5}$$

Based on (D.4) and (D.5), we can obtain the unique solution for (D.3):

$$\begin{aligned} & d^\pi(s_t, a_t) \\ &= \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} d^\pi(s_{t-1}, a_{t-1}) P(s_t|s_{t-1}, a_{t-1})\pi(a_{t-1}|s_{t-1}) \\ &= \sum_{(s_{t-1},a_{t-1})\in\mathcal{Q}_{t-1}} \sum_{(s_{t-2},a_{t-2})\in\mathcal{Q}_{t-2}} d^\pi(s_{t-2}, a_{t-2}) P(s_{t-1}|s_{t-2}, a_{t-2})\pi(a_{t-1}|s_{t-1}) P(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t) \\ &\cdots \\ &= \frac{1}{T_{\max}} \sum_{\{(s_k,a_k)\}_{k=1}^{t-1}} [\mu_0(s_1)\pi(a_1|s_1) P(s_2|a_1, s_1) \cdots P(s_t|a_{t-1}, s_{t-1})\pi(a_t|s_t)], \end{aligned}$$

where we omit the constraint of $\mathcal{Q}_t$ as the transition kernel P naturally satisfies the constraints.

Till now, we have shown that the augmented MDP has a unique stationary state-action visitation distribution shown in (D.1) (the first two points of the Theorem).

Next, we show that the policy value of the policy π under the augmented MDP is proportional to its counterpart under the original MDP without the augmentation (the third point of the Theorem).

Recall that the expected reward of original MDP (5.1) is defined as

$$\begin{aligned}
&\rho(\pi) = \mathbb{E}_{h\sim\mu_0,\pi,\mathcal{E}}[R(s_T, a_T)] = \sum_{T=1}^{T_{\max}} \sum_{h} Pr(h, h \text{ has } T \text{ turns}) R(s_T, a_T) \\
&= \sum_{T=1}^{T_{\max}} \sum_{\{(s_k,a_k)\}_{k=1}^{T-1}} \mu_0(s_1)\pi(a_1|s_1)P(s_2|a_1,s_1)\cdots P(s_T|a_{T-1},s_{T-1})\pi(a_T|s_T) \\
&\times \mathbb{1}(a_T \text{ End Conversation}) R(s_T, a_T),
\end{aligned}$$

where T is the number of turns in the original dialog before padding. Recall that, the MDP only obtain non-zero reward when the dialog ends, (i.e., when a End Conversation). On the other hand, Due to the existence of unique stationary distribution, the policy value of π for the augmented MDP (5.2) can written as:

$$\begin{aligned}
\rho_A(\pi) &= \mathbb{E}_{(s,a)\sim d^\pi(s,a)}[R(s,a)] = \mathbb{E}_{(s,a)\sim d^\pi(s,a)}[\mathbb{1}(a \text{ End Conversation}) R(s,a)] \\
&= \frac{1}{T_{\max}} \sum_{t=1}^{T_{\max}} \sum_{\{(s_k,a_k)\}_{k=1}^{t}} \mu_0(s_1)\pi(a_1|s_1)P(s_2|a_1,s_1)\cdots P(s_t|a_{t-1},s_{t-1})\pi(a_t|s_t) \\
&\quad \times \mathbb{1}(a_t \text{ End Conversation}) R(s_t, a_t) \\
&= \frac{1}{T_{\max}} \rho(\pi).
\end{aligned}$$

□

Can we directly apply infinite-horizon augmentation without padding? The answer

is *NO*. Here we use an example to illustrate the difference between ρ_{A} and ρ and why we need to pad every dialog to have the same length for using OPE:

Example 1. Suppose you have two experience dialogs $a_0 \to \cdots \to a_{t_1}$ and $b_0 \to \cdots \to b_{t_2}$ with rewards 0 and 1 respectively. For the target policy, dialogs has per-episode density 0.2 and 0.8 respectively. The true value of such policy is $0 \times 0.2 + 1 \times 0.8 = 0.8$. The corresponding per-state density of $[a_0, \cdots, a_{t_1}]$ is $\frac{0.2}{0.2\times t_1 + 0.8 \times t_2}$ and the one for $[b_0, \cdots, b_{t_1}]$ is $\frac{0.8}{0.2\times t_1 + 0.8 \times t_2}$. The value in the new augmented MDP is $\frac{0.2*0+0.8*1}{0.2\times t_1 + 0.8 \times t_2}$, which depends on the dialog turns and can not be directly turned into policy value in the original MDP.

D.1.2 Function Approximation with Pre-Trained Language Models

We can compute all state-action pairs for the same dialog in a parallel way as shown in Figure D.1. The input to the RoBERTa encoder consists of three parts, word tokens, position ids, and token types.

Notation: an experience dialog $h = \{e_0, a_1, e_1, ..., a_T\}$, and the corresponding response generated by the target policy π, $\{a'_t = \pi(s_t)\}_{t=1}^{T}$.

Word Tokens. The input token is the concatenation of responses $\{e_0, a_1, a'_1, e_1, ..., e_{T-1}, a_T, a'_T\}$.

Position Ids. The position ids is separately calculated for each response. For e_i, the position ids is from $l_{2i} = \sum_{j<i} \mathrm{len}(e_j) + \sum_{j \le i} \mathrm{len}(a_j)$ to $l_{2i+1} = l_{2i} + \mathrm{len}(e_i)$, where $\mathrm{len}(\cdot)$ denotes the number of tokens of a given response. For a_i, the position ids is from l_{2i-1} to l_{2i}. For a'_i, the position ids is from l_{2i-1} to $l'_{2i} = l_{2i-1} + \mathrm{len}(a'_i)$.

Token types. For e_i's, the token types are 0 which denotes human responses. For a_i's and a'_i's, the token types are 1 which denotes agent responses.

Attention Masking. We need to modify the attention masks to prevent tokens from attending future responses. Specifically, the attention masks make sure:

1. The tokens in each response can be mutually attended;

2. e_i attends to $\{e_0, a_1, e_1, a_2, ..., e_{i-1}, a_i\}$;

3. a_i attends to $\{e_0, a_1, e_1, a_2, ..., a_{i-1}, e_{i-1}\}$;

4. a'_i attends to $\{e_0, a_1, e_1, a_2, ..., a_{i-1}, e_{i-1}\}$;

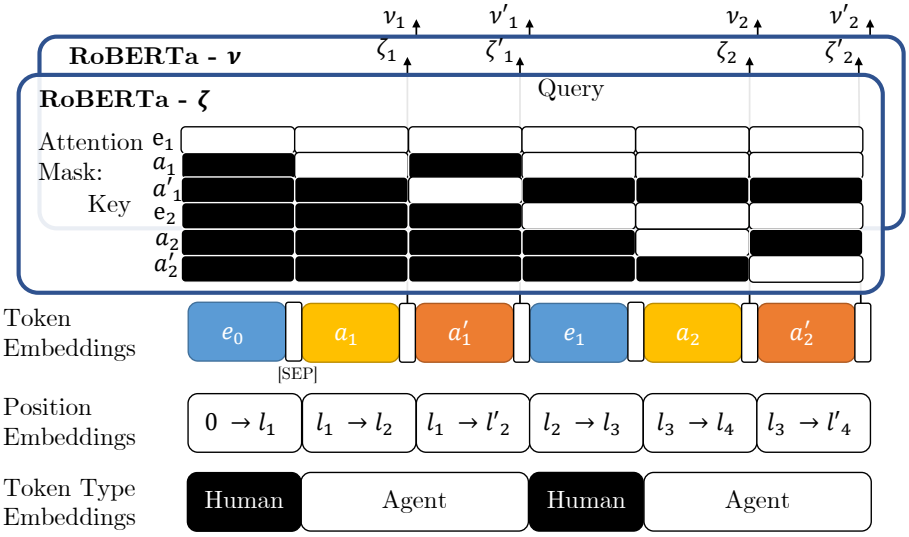


Figure D.1: RoBERTa-ζ and RoBERTa-Q

D.1.3 ENIGMA with regularized DICE

Algorithm 5 Dialog OPE using regularized DICE

Input: Experience Dialog with rewards $\mathcal{D} = \{(h_i = \{e_0^{(i)}, a_1^{(i)}, e_1^{(i)}, ..., a_{T_i}^{(i)}\}, r^{(i)})\}_{i=1}^N$, Target Policy π, Padding Length $T_{\max}$, Regularization function f, DICE hyper-parameters α_ζ, α_R

Output: Performance Estimation $\widehat{\rho}_n(\pi)$

Parameters: $\zeta = \{$ RoBERTa-$\zeta, [\zeta_{\text{pad},t}]_{1\le t\le T_{\max}}\}$,$Q = \{$ RoBERTa-$Q, [Q_{\text{pad},t}]_{1\le t\le T_{\max}}\}$, λ

Generate OPE Data

for $(h_i, r^{(i)})$ in $\mathcal{D}$ **do**
  for t in $1, \cdots, T_i$ **do**
    $\widetilde{a}_t^{(i)} \sim \pi(\{e_0^{(i)}, a_1^{(i)}, e_1^{(i)}, ..., e_{t-1}^{(i)}\})$ // Sample Action From Target Policy
  end for
end for
$\widetilde{\mathcal{D}} = \{(\widetilde{h}_i = e_0^{(i)}, a_1^{(i)}, e_1^{(i)}, ..., a_{T_i}^{(i)}\}, r^{(i)})\}_{i=1}^N$

Estimate ζ by Regularized DICE

while Not Converged **do**
  Sample Mini-Batch $\mathcal{B} \subset \widetilde{\mathcal{D}}$
  for $(\widetilde{h}_i, r^{(i)})$ in $\mathcal{B}$ **do**
    $\zeta_0^{(i)} = \zeta_{\text{pad},T_{\max}}$, $Q_0^{(i)} = Q_0'^{(i)} = Q_{\text{pad},T_{\max}}$ // infinite-horizon concatenation
    for t in $1, \cdots, T_i$ **do**
      $\zeta_t^{(i)} =$ RoBERTa-$\zeta(e_0^{(i)}, a_1^{(i)}, ..., a_{t-1}^{(i)}, e_{t-1}^{(i)}, a_t^{(i)})$
      $Q_t^{(i)} =$ RoBERTa-$Q(e_0^{(i)}, a_1^{(i)}, ..., a_{t-1}^{(i)}, e_{t-1}^{(i)}, a_t^{(i)})$
      $Q_t'^{(i)} =$ RoBERTa-$Q(e_0^{(i)}, a_1^{(i)}, ..., a_{t-1}^{(i)}, e_{t-1}^{(i)}, \widetilde{a}_t^{(i)})$
    end for
    for t in $T_i + 1, \cdots, T_{\max}$ **do**
      $\zeta_t^{(i)} = \zeta_{\text{pad},t}$, $Q_t^{(i)} = Q_t'^{(i)} = Q_{\text{pad},t}$ // add padding state
    end for
    $\ell_i = \frac{1}{T_{\max}}[\alpha_R \zeta_{T_i}^{(i)} r_{T_i}^{(i)} + \sum_{t=0}^{T_{\max}-1}[\zeta_t(Q_{t+1}'^{(i)} - Q_t^{(i)}) + \lambda(\zeta_t^{(i)} - 1) - \alpha_\zeta f(\zeta_t^{(i)})]]$
  end for
  $L_D(\zeta, Q, \lambda) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \ell_i$
  SGD update based on $\frac{\partial L_D}{\partial Q}, \frac{\partial L_D}{\partial \lambda}, \frac{\partial -L_D}{\partial \zeta}$ (Gradient Reversal).
end while

Estimate Average Per-Episode Reward with ζ-Normalization

for $(h_i, r^{(i)})$ in $\mathcal{D}$ **do**
  $\zeta_i =$ RoBERTa-$\zeta(e_0^{(i)}, a_1^{(i)}, ..., a_{T_i-1}^{(i)}, e_{T_i-1}^{(i)}, a_{T_i}^{(i)})$
end for
Return $\widehat{\rho}_n(\pi) = \sum_{(h_i, r^{(i)})\in\mathcal{D}} \zeta_i r^{(i)} \big/ \sum_{(h_i, r^{(i)})\in\mathcal{D}} \zeta_i$

D.2 Experiment Set-Up

In the following experiments, we share the RoBERTa encoder for RoBERTa-ζ and RoBERTa-ν. On the top of RoBERTa-ζ and RoBERTa-ν, it is a two-layer fully connected neural network equipped with GeLU activation [204] and the same hidden dimension as RoBERTa. The RoBERTa encoder is initialized from RoBERTa-base checkpoint [16]. We simply use reverse gradients for the mini-max updates. We set learning rate as 2×10^{-4} and use inverse square root learning rate decay. We impose the gradient norm clipping with the maximum norm $\|\cdot\|_2 \leq 10$. We use 100 times larger learning rate for optimizing λ, 2 times larger learning rate for RoBERTa-Q. In (5.6), we set $\alpha_\zeta = 1$, $f(x) = x^2$ as suggested in Yang et al. [173]. We maintain $\zeta \geq 0$ by adding a square activation at the end of RoBERTa-ζ. The source code is built based on Transformers [205], AirDialog [4], and ParlAI [206]. All experiments are conducted on a machine with $8\times$ V100 GPUs on Google Cloud.

D.3 Transformer-Based Agents for AirDialog

Seller Agent Transformer Architecture There are four components for the encoder: ticket encoder, reservation encoder, dialog encoder, and task-specific heads (intent classification head and name classification head). All tickets and reservation are converted to natural languages. Noticing that, we always append a pseudo ticket in the ticket database representing "no ticket found" situation. The architecture is illustrated in Figure D.2.

Customer Agent Transformer Architecture There are two components for the encoder: intent encoder, reservation encoder. All intents are converted to natural languages. The architecture is illustrated in Figure D.3.

Training Objective Besides the language generation loss $\mathcal{L}_l$, the training objective for seller consists of three parts: name loss, flight loss, intent loss:

$$\min_\theta \mathcal{L}_s(\theta) = \mathcal{L}_l(\theta) + \lambda_n \mathcal{L}_{\text{name}}(\theta) + \lambda_f \mathcal{L}_{\text{flight}}(\theta) + \lambda_i \mathcal{L}_{\text{intent}}(\theta) \tag{D.6}$$

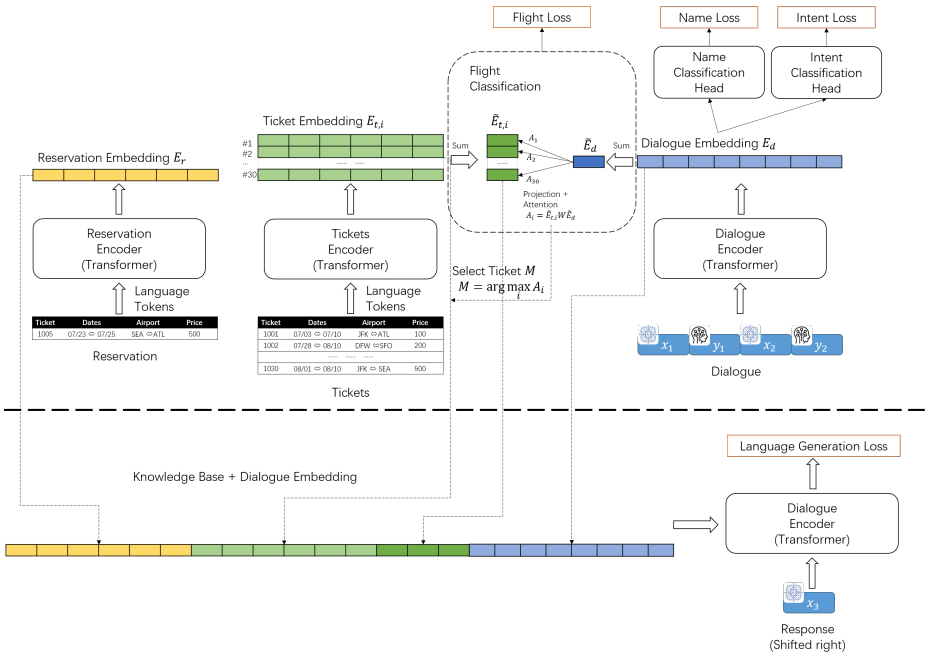


Figure D.2: Transformer-based Seller Agent

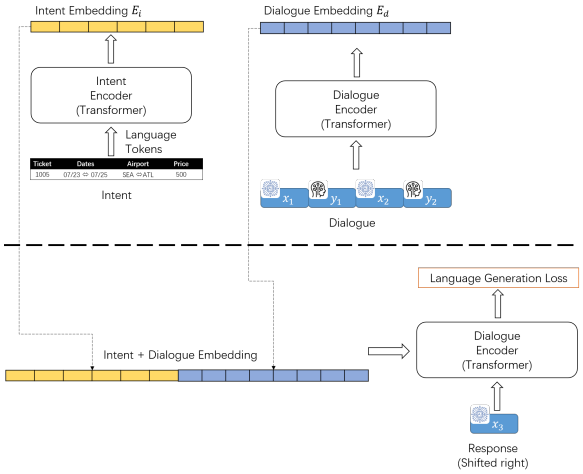


Figure D.3: Transformer-based Customer Agent

The customer agent is trained with normal language generation loss.

Benchmark We compare the proposed model with the current AirDialog RNN baseline [4]. As can be seen, the agent used in this paper are significantly stronger than the baseline agent used in Wei et al. [4].

Hyper-Parameters For training 24 seller agents used in Section 5.4, we varies the size of training data (number of training dialogs) from $5K$ to the full size and varies λ_i and λ_f

Table D.1: Benchmark of the proposed transformer based agent. 'C' means customer, 'S' means seller. Reward, Name, Flight, status are the task-specific scores obtained from self-play evaluation.

Model	BLEU (C)	BLEU (S)	PPL (C)	PPL (S)	Reward	Name	Flight	Status
RNN	22.92	32.95	-	-	0.23	0.41	0.13	0.29
Ours	31.78	31.70	1.671	1.843	0.702	1.00	0.547	0.761

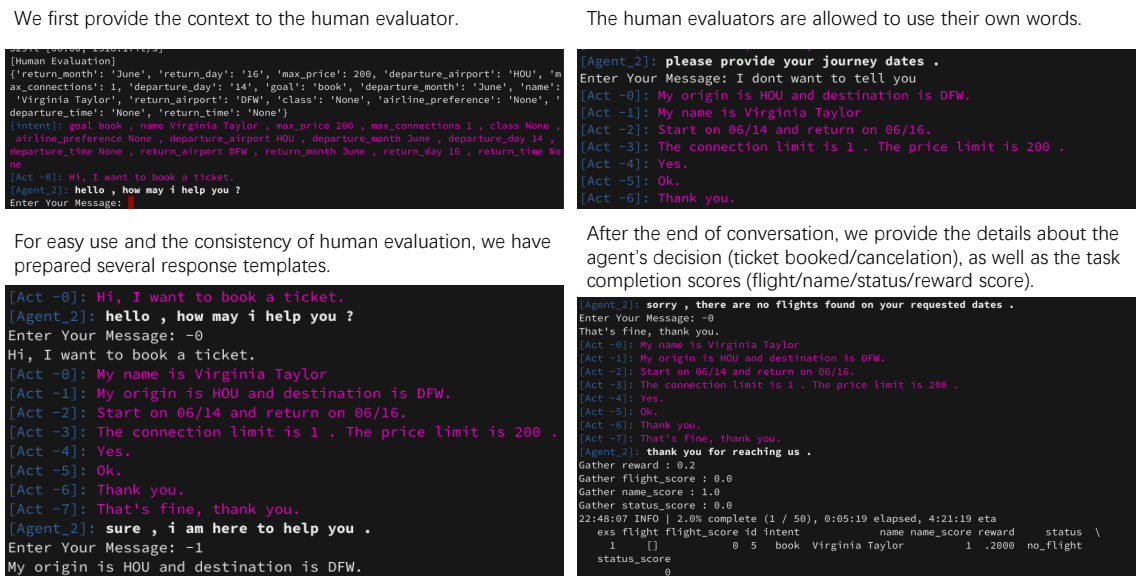


Figure D.4: Screen Shots of Human Evaluation Software

from 0.0001 to 1. For training the customer agent used in self-play evaluation, we use the full training data and tune the hyperparameters based on the BLEU score evaluated using the validation set.

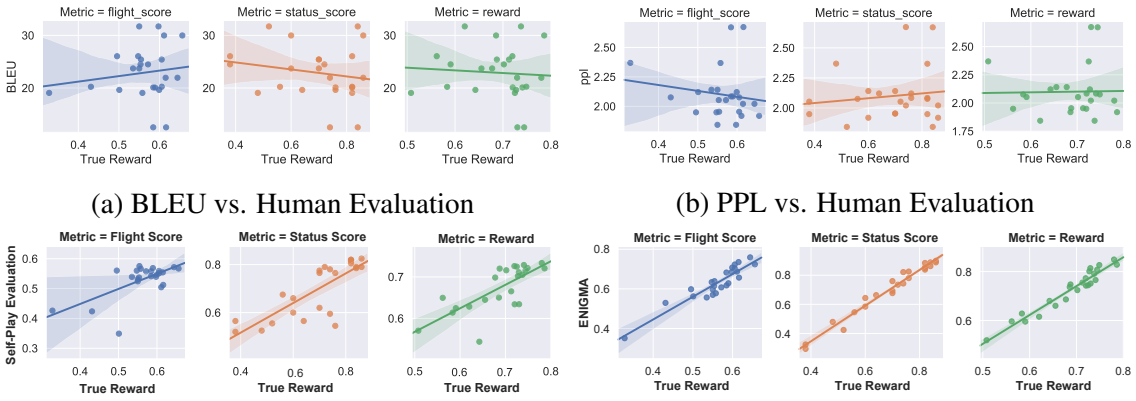
Human Evaluation The human evaluation is collected from 20 different Ph.D. students majored in Math/Stats/CS/IEOR. We provide detailed guidelines to the human evaluator that they have to speak to the agents with similar tone. Figure D.4 presents the screen shots of the human evaluation software.

D.4 Additional Experiment

D.4.1 AirDialog

Regression Plot

We present the regression plot for the full setting in Figure D.5 and for the selected agent in Figure D.6.



(a) BLEU vs. Human Evaluation

(b) PPL vs. Human Evaluation

(c) SPE vs. Human Evaluation

(d) ENIGMA vs. Human Evaluation

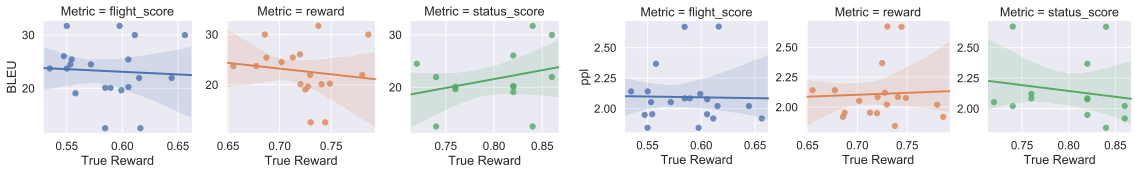
Figure D.5: Regression Plot. The x-axis is the average reward obtained by chatting with human. The y-axis is BLEU/PPL/the reward estimated by ENIGMA. Different colors denotes different type of rewards (flight score, status score, and overall reward). The solid line is obtained by linear regression and the shaded region indicates 95% confidence interval. (see more in *seaborn* packages).

Training Curves

We show the training curves of the ENIGMA in Figure D.7. Here four models are presented, the best model (ranked 100%), model ranked as 50%, model ranked as 25% and the worst model (ranked 0%). As can been seen the estimated reward estimation converges steadily to it's true values.

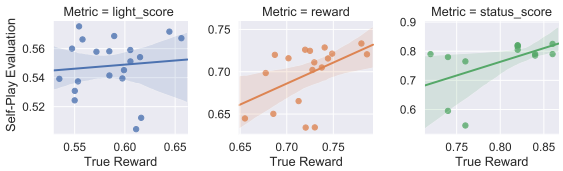
Ablation Study

Here we provide large figures (Figure D.8 and Figure D.9) for the ablation study mentioned in Section 5.4.



(a) BLEU vs. Human Evaluation

(b) PPL vs. Human Evaluation



(c) SPE vs. Human Evaluation

(d) ENIGMA vs. Human Evaluation

Figure D.6: Regression Plot for "selected agent" Setting. The x-axis is the average reward obtained by chatting with human. The y-axis is BLEU/PPL/the reward estimated by ENIGMA/Self-Play Evaluation (SPE). Different colors denotes different type of rewards (flight score, status score, and overall reward). The solid line is obtained by linear regression and the shaded region indicates 95% confidence interval. (see more in *seaborn* packages).

D.4.2 Additional Results for Rule-Based Agents of AirDialog

• *Rule-Rule (R-R)*: Both customer and seller agents are rule based. We fix the customer rule-based model and construct and evaluate 6 seller agents. The strongest agent can perfectly interpret the intent of rule-based customers. While the weaker agents interprets the intent with different levels of noise. The learning curve is presented in Figure D.10.

Table D.2: The correlation between two metrics. Each column is a task completion score obtained by interacting with the environments under R-R setting. Each row is an automatic metric.

Setting	**Method**	**Pearson Correlation**			**Spearman's Rank Correlation**		
		Flight Score	**Status Score**	**Reward**	**Flight Score**	**Status Score**	**Reward**
R-R	BLEU	0.1981	-0.0067	0.0980	0.1525	0.0009	0.0924
	ppl	-0.1584	-0.0610	-0.1209	-0.2475	-0.1060	-0.1178
	ENIGMA	**0.9687**	**0.9947**	**0.9874**	**0.8800**	**0.9872**	**0.9574**

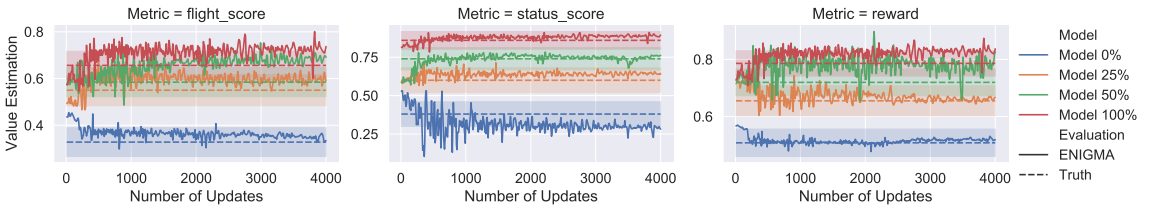


Figure D.7: Learning curve for AirDialog. The x-axis is the number of mini-max updates, while y-axis is the estimated values. The straight line is the true reward, while the shaded region denotes the 90% confidence interval. The true reward and the confidence interval is obtained via different evaluation chats between the agents and the environment (model/human). Different colors denotes different agents.

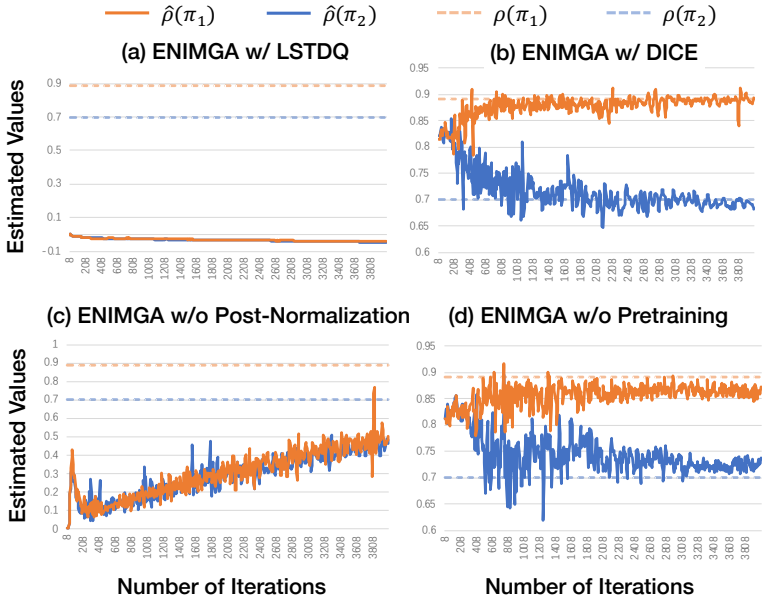


Figure D.8: Reward estimation of two target agents (π_1 and π_2) vs. # of iterations. Dotted lines represents true rewards.

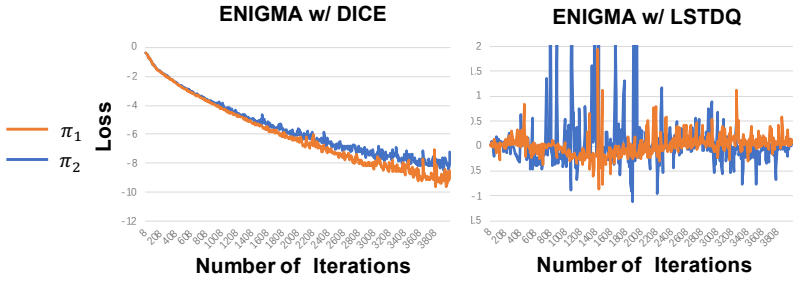


Figure D.9: Loss value of two target agents during mini-max optimization.

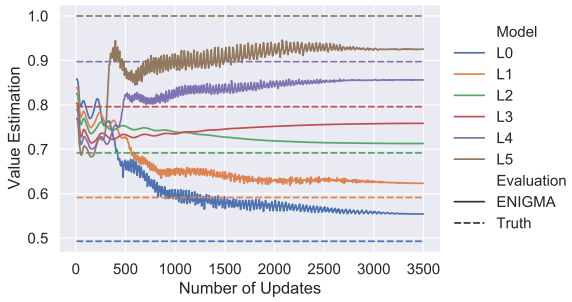


Figure D.10: Learning Curve under Rule-Rule setting

D.4.3 ConvAI2

Training Curves.

Similar to the AirDialog dataset, we also show the training curves for the agents ranked at 100%, 50%, 25%, 0% in Figure D.11. ENIGMA also converges steadily to the true values within a resonable error.

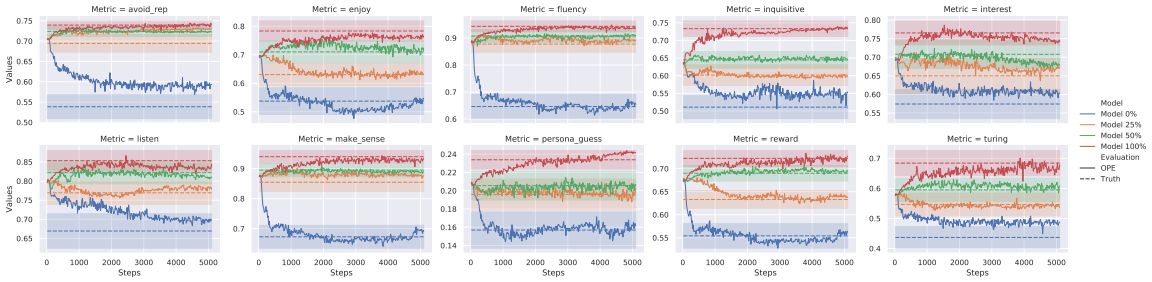


Figure D.11: Learning curve for ConvAI2. The x-axis is the number of mini-max updates, while y-axis is estimated values. The straight line is the true reward, while the shaded area denotes the 95% confidence interval. The true reward and the confidence interval is obtained via different evaluation chats between the agents and human. Different colors denotes different agents.

Regression Plot.

We present the regression plot for the all 10 metrics in setting in Figure D.12. The corresponding corresponding correlation is presented in Table D.3. For comparison, we present the regression plot for self-play evaluation in Figure D.13.

Experience Data. To analysis how many human-model evaluation dialogs are needed, we analysis ENIGMA error under different sizes of the experience data. For ConvAI2, we compare the error for using 100% data, 50% data and 10% data. As shown in Table D.3 and Figure D.14, when we use half of the data, the error is similar to the one using full data. If we only use 10% data, ENIGMA becomes very inaccurate. OPE under low resource setting remains very challenging.

In Figure D.14, we study the estimation error under different sizes of the experience data. As can be seen, when using 50% data, the reward value estimation is very similar to

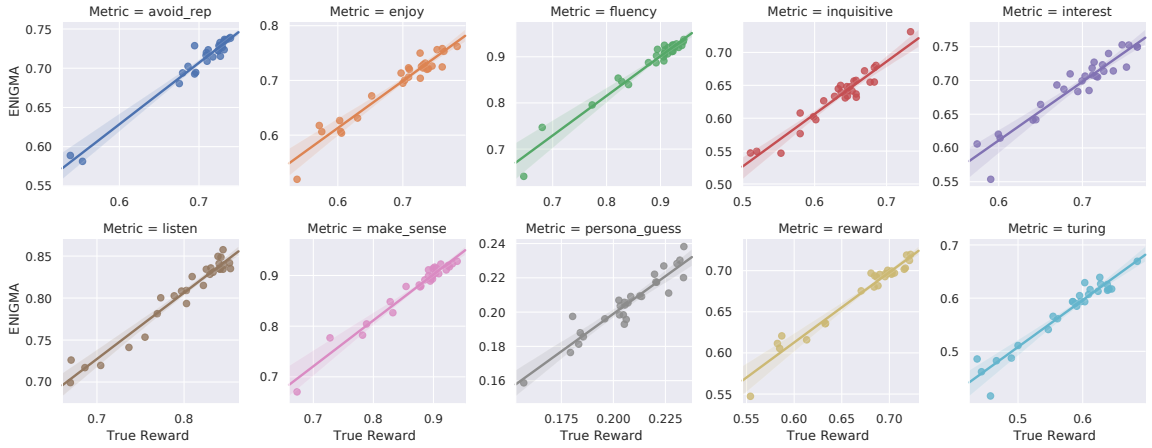


Figure D.12: ENIGMA vs. Human Evaluation for ConvAI2. The x-axis is the average reward obtained by chatting with human. The y-axis it the reward estimated by ENIGMA. Different colors represent different language quality metrics. The solid line is obtained by simple linear regression.

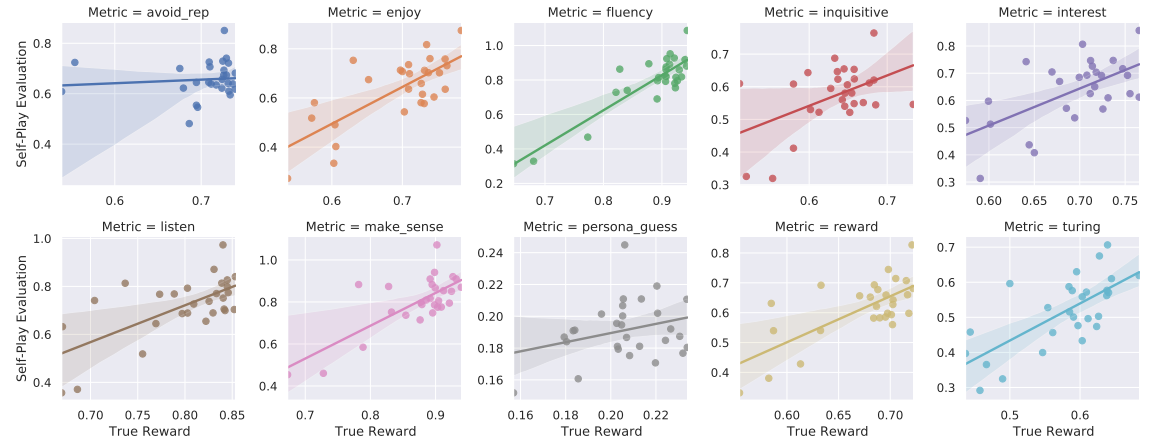


Figure D.13: Self-Play Evaluation vs. Human Evaluation for ConvAI2. The x-axis is the average reward obtained by chatting with human. The y-axis it the reward estimated by self-play evaluation Different colors represent different language quality metrics. The solid line is obtained by simple linear regression.

the one of using full data. When using only 10% data, the error is larger and ENIGMA has lower correlation with the true reward.

A More Challenging Setting.

Considering that some target agents are similar to the behavior policies with only slight difference in the way of decoding, they might yield very the similar dialog when the human

Table D.3: The correlation between different metrics and ENIGMA estimation. Each column is each average language quality score obtained by chatting with human. Different rows represent different experience data ENIGMA used.

Pearson Correlation					
Setting	**Avoid Rep.**	**Enjoy**	**Fluency**	**Inquisitive**	**Interest**
Full Data	0.9792	0.9661	0.9767	0.9584	0.9488
50% Data	0.9573	0.9046	0.9550	0.9237	0.8644
10% Data	0.9266	0.6595	0.8910	0.8286	0.5052
Selected Data	0.6944	0.6759	0.7762	0.5605	0.5820
Setting	**Listen**	**Make Sense**	**Persona**	**Reward**	**Turing**
Full Data	0.9754	0.9788	0.9415	0.9773	0.9637
50% Data	0.8971	0.9585	0.8770	0.9374	0.8506
10% Data	0.7455	0.8100	0.4544	0.8240	0.6825
Selected Data	0.5520	0.7402	0.6879	0.6968	0.5394
Spearman's rank correlation					
Setting	**Avoid Rep.**	**Enjoy**	**Fluency**	**Inquisitive**	**Interest**
Full Data	0.8905	0.9070	0.9178	0.8717	0.9210
50% Data	0.7558	0.7980	0.6651	0.8482	0.7727
10% Data	0.4128	0.6147	0.6492	0.6335	0.4713
Selected Data	0.5138	0.5561	0.4522	0.3168	0.6027
Setting	**Listen**	**Make Sense**	**Persona**	**Reward**	**Turing**
Full Data	0.9240	0.9448	0.9205	0.9485	0.9213
50% Data	0.7784	0.8647	0.8293	0.7750	0.7026
10% Data	0.3914	0.5096	0.3651	0.5774	0.5893
Selected Data	0.4585	0.6126	0.5844	0.6672	0.4259

acts in the same way. Specifically, in the data collection process, the target model might yield the responses that are very similar to the ones of the behavior policy for all turns in the dialog: $\text{EditDistance}(a_t, a'_t) \leq 15 \ \ \forall 0 \leq t \leq T$. For a more realistic setting, we consider removing these highly overlapped dialogs after the data collection process. This setting is very challenging that the target policy behavior is less covered by the experience data and ENIGMA can only hopefully generalize via pre-trained RoBERTa. The results are shown in Figure D.15 and Table D.3. As can be seen, this setting remains challenging as the Pearson correlation is between 0.5 and 0.8. For comparison, we present the regression plot for self-play evaluation using this challenging subset of the experience data in Figure D.16.

Figure D.14: Error Analysis on Convai2 under different data size. The x-axis is the true average reward. The y-axis is the ENIGMA error. The solid line is the fitted quadratic function. Blue, orange, green colors represent 100%, 50%, 10% datasets respectively.

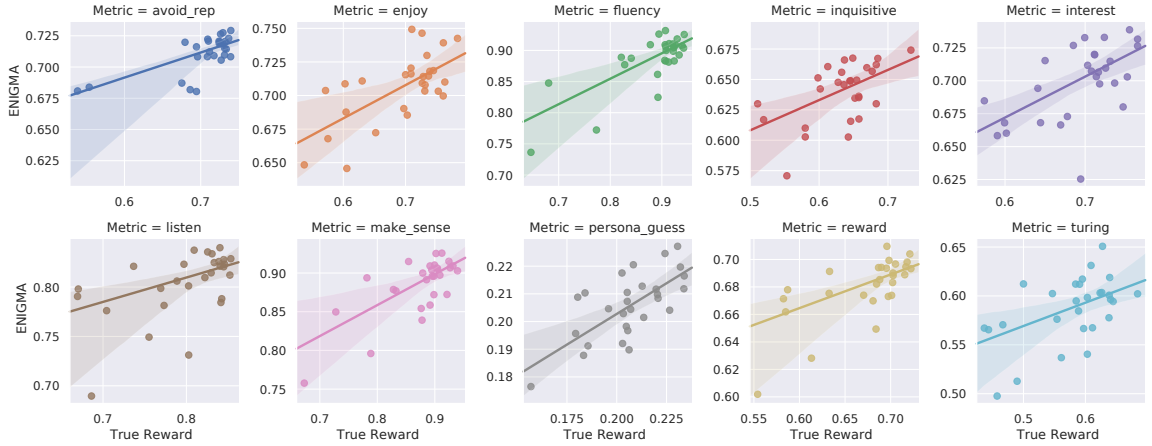


Figure D.15: ENIGMA vs. Human Evaluation for ConvAI2 under the challenging setting. The x-axis is the average reward obtained by chatting with human. The y-axis it the reward estimated by ENIGMA. Different colors represent different language quality metrics. The solid line is obtained by simple linear regression.

We remark that such experiments can also be done for AirDialog. However, due to the limitation that most agents are just learning template responses due to the goal-oriented nature, removing overlapped dialogs results in an extremely incomplete experience dataset. For example, most "cancelation" dialogs will be removed since they are very simple and basically the same for different agents. As a result ENIGMA can not make a reasonable estimation due to the highly incomplete experience data.

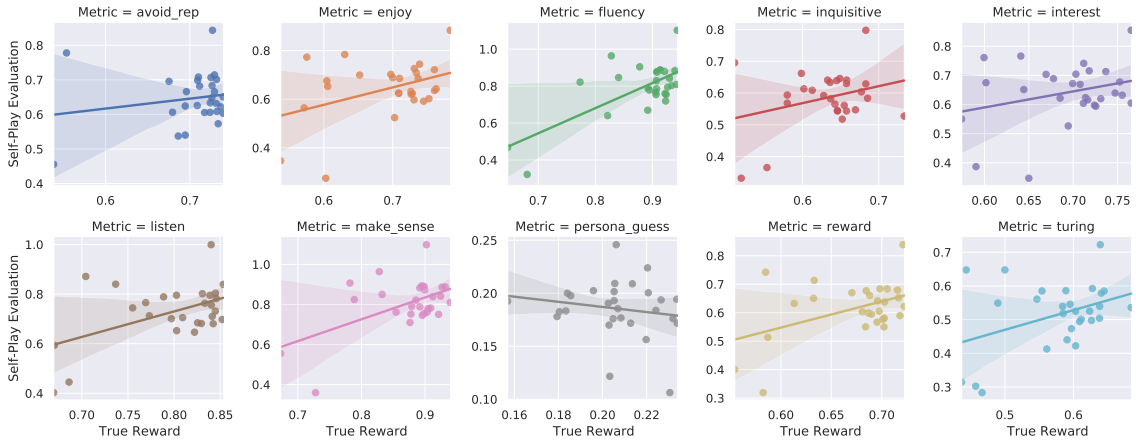


Figure D.16: Self-Play Evaluation vs. Human Evaluation for ConvAI2 under the challenging setting. The x-axis is the average reward obtained by chatting with human. The y-axis it the reward estimated by self-play evaluation. Different colors represent different language quality metrics. The solid line is obtained by simple linear regression.

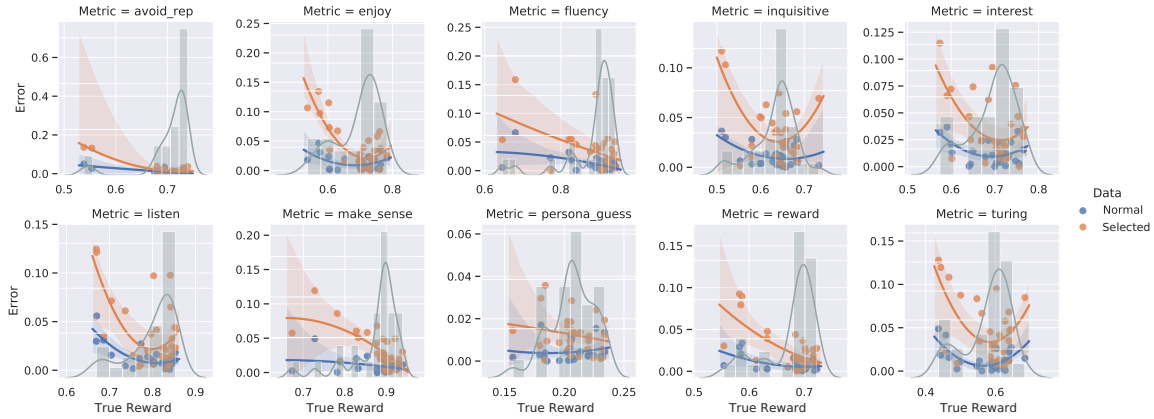


Figure D.17: ENIGMA Error Comparison between using normal and selected challenging experience data on ConvAI2. The x-axis is the true average reward. The y-axis is the ENIGMA error. The solid line is the fitted quadratic function. The histogram is the empirical distribution of the rewards of all the experience data. Orange represents challenging dataset, and blue represents normal dataset.

Figure D.17 compares the error of ENIGMA between using the normal experience data and the selected challenging one. As can be seen, the error using the selected data is larger particularly for the agents with exceptionally low/high true reward. That indicates the problem of the lack of dialog coverage is exaggerated under the challenging setting, while the ENIGMA estimation remains accurate when there is sufficient dialog coverage.

Comparison to Automatic Hand-crafted Metrics.

We compare ENIGMA with other automatic hand-crafted metrics proposed in See et al. [40]. For a more intuitive comparison, we use heat map and box plot to visualize the correlations between different automatic evaluation metrics and different human evaluation metrics. As can be seen in Figure D.18 and Figure D.19, most hand-crafted metrics have relatively low correlation to human evaluation metrics. The only exception is the "question marks" automatic metrics for inquisitive human evaluation metric. Some hand-crafted metrics have high Pearson correlation to some human evaluation metrics, while the corresponding Spearman's rank correlation is low. The reason is that they can easily identify some extremely good/bad agents while they are less effective for identifying agents with similar performance.

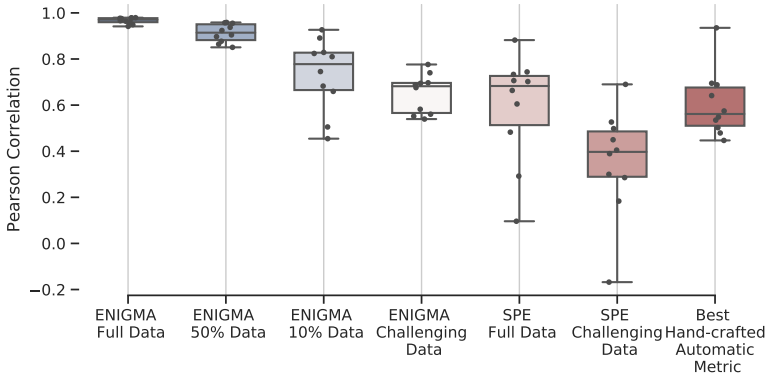
	Avoid Rep.	Enjoy	Fluency	Inquisitive	Interest	Listen	Make Sense	Persona Guess	Reward	Turing
Repetition External Bigram	0.94	0.50	0.17	0.22	0.55	0.53	0.08	0.03	0.48	0.57
Repetition External Unigram	0.93	0.49	0.18	0.29	0.54	0.52	0.07	0.03	0.48	0.56
Repetition Internal Bigram	0.35	0.40	0.15	0.38	0.48	0.25	0.16	0.09	0.34	0.30
Repetition Internal Unigram	0.65	0.33	0.13	0.45	0.34	0.31	0.10	0.14	0.35	0.40
Repetition Partner Rep. Bigram	0.33	0.16	0.42	0.34	0.03	0.16	0.51	0.11	0.22	0.08
Specificity	0.05	0.29	0.69	0.52	0.09	0.47	0.64	0.18	0.43	0.34
Response-rel	0.02	0.29	0.54	0.26	0.16	0.17	0.45	0.45	0.33	0.22
Questions	0.16	0.17	0.26	0.69	0.11	0.13	0.34	0.26	0.21	0.06
8 Features + Linear Regression	0.74	0.58	0.59	0.75	0.54	0.64	0.44	-0.03	0.61	0.53
SPE Full Data	0.73	0.60	0.66	0.70	0.10	0.74	0.88	0.29	0.48	0.71
SPE Challenging Data	0.40	0.29	0.50	0.39	0.18	0.53	0.69	-0.17	0.30	0.45
ENIGMA Full Data	0.98	0.97	0.98	0.96	0.95	0.98	0.98	0.94	0.98	0.96
ENIGMA 50% Data	0.96	0.90	0.96	0.92	0.86	0.90	0.96	0.88	0.94	0.85
ENIGMA 10% Data	0.93	0.66	0.89	0.83	0.51	0.75	0.81	0.45	0.82	0.68
ENIGMA Challenging Data	0.69	0.68	0.78	0.56	0.58	0.55	0.74	0.69	0.70	0.54

(a) Pearson Correlation

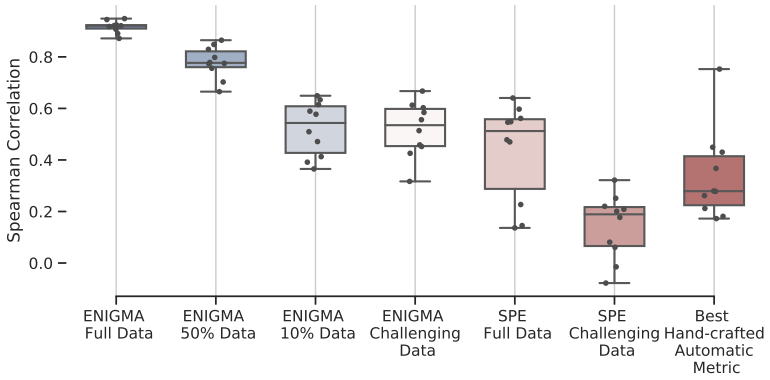
	Avoid Rep.	Enjoy	Fluency	Inquisitive	Interest	Listen	Make Sense	Persona Guess	Reward	Turing
Repetition External Bigram	0.26	0.07	0.18	0.42	0.06	0.02	0.25	0.40	0.02	0.07
Repetition External Unigram	0.27	0.21	0.12	0.07	0.25	0.03	0.06	0.12	0.17	0.28
Repetition Internal Bigram	0.17	0.00	0.12	0.10	0.09	0.18	0.28	0.00	0.04	0.03
Repetition Internal Unigram	0.22	0.15	0.07	0.07	0.19	0.10	0.11	0.07	0.11	0.22
Repetition Partner Rep. Bigram	0.37	0.12	0.12	0.27	0.18	0.01	0.22	0.18	0.08	0.13
Specificity	0.30	0.11	0.23	0.33	0.18	0.06	0.28	0.43	0.03	0.09
Response-rel	0.26	0.17	0.07	0.08	0.26	0.07	0.15	0.22	0.11	0.15
Questions	0.18	0.13	0.28	0.75	0.13	0.18	0.45	0.37	0.17	0.03
8 Features + Linear Regression	0.48	0.45	0.52	0.79	0.28	0.48	0.53	0.35	0.45	0.42
SPE Full Data	0.60	0.47	0.55	0.64	0.14	0.48	0.56	0.15	0.23	0.55
SPE Challenging Data	0.21	0.08	0.25	0.20	0.06	0.18	0.32	-0.08	-0.01	0.22
ENIGMA Full Data	0.89	0.91	0.92	0.87	0.92	0.92	0.94	0.92	0.95	0.92
ENIGMA 50% Data	0.76	0.80	0.67	0.85	0.77	0.78	0.86	0.83	0.78	0.70
ENIGMA 10% Data	0.41	0.61	0.65	0.63	0.47	0.39	0.51	0.37	0.58	0.59
ENIGMA Challenging Data	0.51	0.56	0.45	0.32	0.60	0.46	0.61	0.58	0.67	0.43

(b) Spearman's Rank Correlation

Figure D.18: Heat map for correlation between different automatic evaluation metrics and different human evaluation metrics. Different rows represent different automatic metrics. Different column represent different human evaluation metrics.



(a) Pearson Correlation

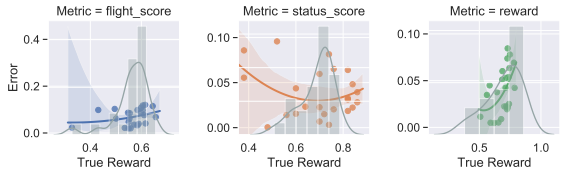


(b) Spearman's Rank Correlation

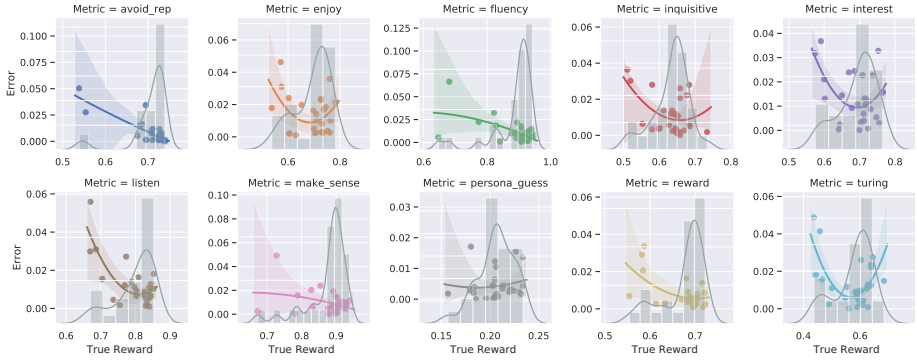
Figure D.19: Box plot of performance. Each box corresponds to each method. There are 10 points for each box representing correlations to 10 different human evaluation metrics.

D.4.4 Error Analysis

We analyze the detailed errors to identify the error pattern for better understand the limit of ENIGMA. We calculate the absolute difference between the estimation and the true average reward. The results are summarized in Figure D.20. A common pattern we see in ConvAI2 is that, when the true average reward is too high or too low, the ENIGMA becomes less accurate. One possible reason for that is the lack of samples of dialogs with the extreme rewards in the experience data. We empirically verify this conjecture by comparing the the error with the reward distribution in the experience data in Figure D.20. For AirDialog, such pattern is not obvious. That is because the quality of the decision module is more important to the agent performance for this task completion scores. As a result, even performance of the target agent is much higher/lower than the experience data, as long as they share similar languages, ENIGMA can estimate the performance accurately.



(a) AirDialog

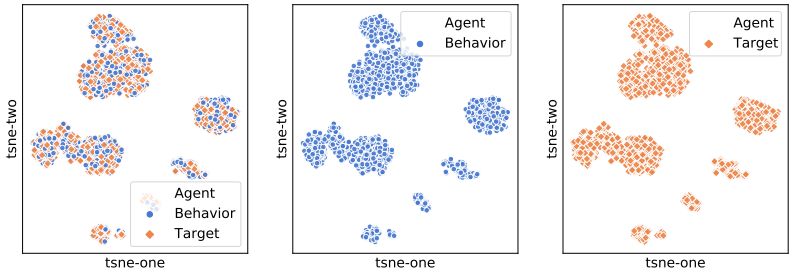


(b) ConvAI2

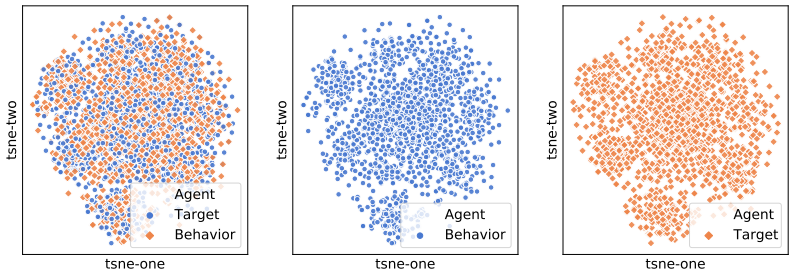
Figure D.20: Error Analysis on AirDialog and ConvAI2. The x-axis is the true reward. The y-axis is the Estimation error. The solid line is the fitted quadratic function. The histogram is the empirical distribution of the true rewards of all the experience data.

D.4.5 Embedding Visualization

In Figure D.21, we present the t-SNE plots for the embedding of the state-action pairs from the behavior experience data and the target policy. The two sets of embeddings provided by the pre-trained language models are largely overlapped with rich semantic information. On the other hand, the embeddings provided by a randomly initialized model spread over the entire high-dimensional space.



(a) Initialized from RoBERTa-base.



(b) Random initialization.

Figure D.21: t-SNE Plots for contextual embedding extracted from RoBERTa-ζ and RoBERTa-ν on AirDialog.

D.5 Automatic Dialog Evaluation Comparison

Table D.4: Comparison between current automatic evaluation approaches. Part of the table is collected from two comprehensive surveys [207, 208]. **Red: Drawback**; **Green: Advantage**.

Method	Criterion	Dynamic (RL)	Model Free	Experience Data	Behavior Policy Similar to Target Policy	Behavior Agnostic	Description / Examples
BLEU, Perplexity,METEOR,ROUGE [184, 209, 210, 211, 212]	Language Quality Score	No	N/A	Human-Human	Yes	N/A	The most widely use metrics: Given a **fixed** dialog history, they compute heuristic scores / statistics based on comparing **single turn** response given by the model and reference human responses. E.g., BLEU, perplexity
Mitchell and Lapata [46], Rus and Lintean [213], Forgues et al. [214], Higashinaka et al. [44], Xiang et al. [43], Wieting et al. [215], Gandhe and Traum [45], Tao et al. [216], Shimanaka, Kajiwara, and Komachi [217], Zhang et al. [218], Ghazarian et al. [219], Li et al. [220], Mehri and Eskenazi [221], Gao et al. [222], Lan et al. [223], Pang et al. [224], Zhang et al. [225], Yuma, Yoshinaga, and Toyoda [226], Zhao, Lala, and Kawahara [227], and Sai et al. [228]	Language Quality Score	No	N/A	Human-human experience data or specially designed data.	Yes (Implicitly) *Although they do not explicitly require such similarity, the single-turn responses of models trained from the same data are usually similar to human responses.*	N/A	Given a **fixed** dialog history, they compute some scores for **single-turn** response given by the model using **an evaluator**, e.g., pretrained word embeddings and pretrained language models. These method are the so-called "embedding-based metrics". The evaluator usually require training on a large-scale text dataset. They may or may not depends on reference human responses. E.g. RUBER [216].
Lowe et al. [41], Huang et al. [229], and Sellam, Das, and Parikh [230]	Language Quality Score	No	N/A	Human-Human and Human-Model	Yes	N/A	Mostly the same as above. In addition, the data for training the evaluator includes human-model experience data to improve performance. E.g., ADEM [41].
Hemphill, Godfrey, and Doddington [231] and Williams et al. [232]	Task Completion Score	No	N/A	Human-Human	No	N/A	They compute task related score of task-specific actions (e.g., intent detection) given by the model for a **fixed complete** dialog. These can only be used to test classification / information retrieval module. E.g., Intent Detection Accuracy.
Wei et al. [4]	Task Completion Score	Yes	No	Human-Human and/or Human-Model	Yes (Implicitly)	N/A	They compute task related score of task-specific actions (e.g., intent detection) given by the model for a dialog that is obtained by **interaction** with a **user simulator**. E.g., Self-Play Evaluation [4].
Ghandeharioun et al. [39]	Language Quality Score	Yes	No	Human-Model	Yes	N/A	Basically the same as above. In addition to modeling human responses, they usually require **modeling human reward function**. E.g., Self-Play Evaluation [39].
Inverse Proportional Score E.g., Horvitz and Thompson [171], Wang, Gao, and Zha [233], and Precup [169] (not practical for dialog)	Both	Yes	Yes	Human-Model	Yes	No (not practical for dialog)	**Directly model the performance** under the **interaction** environment using experience collected from **known** probabilistic models. E.g., Inverse Proportional Score.
ENIGMA	Both	Yes	Yes	Human-Model	Yes	Yes	**Directly model the performance** under **interaction** environment using experience collected from **unknown** distribution. E.g., Q-Learning, ENIGMA.

D.5.1 Static Methods

As can be seen, most previous methods only focus on evaluating *language quality* for **single-turn** response of a **fixed** context. These methods can not evaluate agents under interactive context. As a result, they can not be extended to *goal-oriented* dialogs.

For goal-oriented dialogs, the static evaluation methods are very limited. The static methods can only evaluate the model actions to a **fixed complete** dialog, e.g., intent detection.

D.5.2 Dynamic Methods

Previous dynamic methods under RL framework are based on self-play evluation, which requires learning the environment, i.e, human. As discussed in the main paper, learning a human model is significantly beyond the current technical limit.

ENIGMA overcome learning the environment by directly modeling the performance of agents.

D.5.3 Information Theoretic Limit

The common limitation of all existing methods is that they require similarity between the target policy and behavioral policies, so that the experience data can cover sufficient interaction patterns between the target policy and human.

For example, BLEU score requires the agent response being similar to the reference response. Another example is ADEM [41], they include the target policy into the experience data collection to achieve decent performance (0.37 Pearson correlation to human ratings). If the target policy is excluded from the behavior policies, ADEM only achieves 0.13 Pearson correlation, which is even lower than the one between dialog length and human ratings 0.27.

For static single-turn evaluation for language quality, one might satisfy the requirement by just using human as the behavior policy and large-scale diverse experience data. That is because the single-turn responses of the target model have a very similar pattern to the human responses, as they are usually trained to mimic one-turn human response. However, high similarity of responses between the target model and human requires a very strong target model trained with large-scale data, which is not practical in most settings. Some

existing work try to alleviate such requirement and increase the coverage of experience data by external knowledge graph [229] and synthetic samples [230]. We remark that although the static methods only require single-turn similarity between behavior and target policies, their empirical performance is unsatisfactory comparing with multi-turn interactive human evaluation [39].

In multi-turn interactive evaluation, we can not just use human as the behavior policy especially for goal-oriented dialogs. That is because the multi-turn behavior of the target model is very different from the human behavior. Take Airdialog as an example, human agents can always book the correct tickets while the target model may fail for many times.

Such a limitation is the theoretical requirement of bounded state-action density ratio between target and behavior policies, which has been discussed in many off-policy evaluation literature [234, 180].

Due to such theoretical limitation, a large amount of **human-model** interactive evaluation data is needed to study automatic interactive evaluation. However, most evaluation logs are not publicly available, and research in this direction has largely lagged behind. To the best of our knowledge, ConvAI2 [40] is the only public comprehensive human-model interactive evaluation data. [1] Therefore, we recommend that the research community release human-model interaction evaluation data to promote dialog evaluation/learning research and benefit the entire community.

[1] Our human-model evaluation data on Airdialog will also be released soon.

REFERENCES

Zhang, Lei, Shuai Wang, and Bing Liu (2018). "Deep learning for sentiment analysis: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4, e1253.

Liang, Chen et al. (2020). "Bond: Bert-assisted open-domain named entity recognition with distant supervision". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1054–1064.

Jiang, Haoming et al. (2020b). "SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2177–2190.

Wei, Wei et al. (2018). "Airdialogue: An environment for goal-oriented dialogue research". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3844–3854.

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

Peters, Matthew E et al. (2018). "Deep contextualized word representations". In: *Proceedings of NAACL-HLT*, pp. 2227–2237.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8.

Devlin, Jacob et al. (2019a). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems*, pp. 5998–6008.

Brown, Tom B. et al. (2020). "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165.

Yang, Zhilin et al. (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems*, pp. 5754–5764.

Dong, Li et al. (2019). "Unified language model pre-training for natural language understanding and generation". In: pp. 13042–13054.

Joshi, Mandar et al. (2020). “Spanbert: Improving pre-training by representing and predicting spans”. In: *Transactions of the Association for Computational Linguistics* 8, pp. 64–77.

Lan, Zhenzhong et al. (2019). “ALBERT: A lite BERT for self-supervised learning of language representations”. In: *arXiv preprint arXiv:1909.11942*.

Raffel, Colin et al. (2019a). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *arXiv preprint arXiv:1910.10683*.

Liu, Yinhan et al. (2019c). “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692*.

Liu, Xiaodong et al. (2019a). “Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding”. In: *arXiv preprint arXiv:1904.09482*.

— (July 2019b). “Multi-Task Deep Neural Networks for Natural Language Understanding”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4487–4496.

Howard, Jeremy and Sebastian Ruder (2018). “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339.

Peters, Matthew E, Sebastian Ruder, and Noah A Smith (2019). “To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks”. In: *ACL 2019*, p. 7.

Houlsby, Neil et al. (2019). “Parameter-Efficient Transfer Learning for NLP”. In: *International Conference on Machine Learning*, pp. 2790–2799.

Stickland, Asa Cooper and Iain Murray (2019). “BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning”. In: *International Conference on Machine Learning*, pp. 5986–5995.

Xie, Qizhe et al. (2019). “Unsupervised Data Augmentation for Consistency Training”. In: *CoRR* abs/1904.12848. arXiv: 1904.12848.

Awasthi, Abhijeet et al. (2020). “Learning from Rules Generalizing Labeled Exemplars”. In: *International Conference on Learning Representations*.

Xu, Yige et al. (2020). “Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation”. In: *CoRR* abs/2002.10345. arXiv: 2002.10345.

Zhu, Chen et al. (2020a). “FreeLB: Enhanced Adversarial Training for Natural Language Understanding”. In: *International Conference on Learning Representations*.

Jiang, Haoming et al. (July 2020c). “SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2177–2190.

Ratner, Alexander et al. (2020). “Snorkel: rapid training data creation with weak supervision”. In: *VLDB Journal* 29.2, pp. 709–730.

Varma, Paroma and Christopher Ré (Nov. 2018). “Snuba: Automating Weak Supervision to Label Training Data”. In: *Proc. VLDB Endowment* 12.3, 223–236.

Mallinar, Neil et al. (2019). “Bootstrapping conversational agents with weak supervision”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 9528–9533.

Aina, Laura, Kristina Gulordava, and Gemma Boleda (July 2019). “Putting Words in Context: LSTM Language Models and Lexical Ambiguity”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3342–3348.

Luo, Bingfeng et al. (July 2017). “Learning with Noise: Enhance Distantly Supervised Relation Extraction with Dynamic Transition Matrix”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 430–439.

Wang, Hao et al. (Nov. 2019b). “Learning with Noisy Labels for Sentence-level Sentiment Classification”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6286–6292.

Ren, Wendi et al. (Nov. 2020). “Denoising Multi-Source Weak Supervision for Neural Text Classification”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3739–3754.

Rosenberg, Chuck, Martial Hebert, and Henry Schneiderman (2005). “Semi-supervised self-training of object detection models.” In: *WACV/MOTION* 2.

Lee, Dong-Hyun (2013). “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks”. In: *Workshop on challenges in representation learning, ICML*. Vol. 3, p. 2.

Turing, AM (1950). “Computing Machinery and Intelligence”. In: *Mind* 59.236, pp. 433–460.

Liu, Chia-Wei et al. (2016). “How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation”. In: *arXiv preprint arXiv:1603.08023*.

Ghandeharioun, Asma et al. (2019). “Approximating interactive human evaluation with self-play for open-domain dialog systems”. In: *Advances in Neural Information Processing Systems*, pp. 13658–13669.

See, Abigail et al. (2019). “What makes a good conversation? How controllable attributes affect human judgments”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Lowe, Ryan et al. (2017). “Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1116–1126.

DeVault, David, Anton Leuski, and Kenji Sagae (2011). “Toward learning and evaluation of dialogue policies with text examples”. In: *Proceedings of the SIGDIAL 2011 Conference*, pp. 39–48.

Xiang, Yang et al. (2014). “Problematic situation analysis and automatic recognition for chinese online conversational system”. In: *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pp. 43–51.

Higashinaka, Ryuichiro et al. (2014). “Evaluating coherence in open domain conversational systems”. In: *Fifteenth Annual Conference of the International Speech Communication Association*.

Gandhe, Sudeep and David Traum (2016). “A semi-automated evaluation metric for dialogue model coherence”. In: *Situated Dialog in Speech-Based Human-Computer Interaction*. Springer, pp. 217–225.

Mitchell, Jeff and Mirella Lapata (2008). “Vector-based models of semantic composition”. In: *proceedings of ACL-08: HLT*, pp. 236–244.

Dziri, Nouha et al. (2019). “Evaluating coherence in dialogue systems using entailment”. In: *arXiv preprint arXiv:1904.03371*.

Möller, Sebastian et al. (2006). “MeMo: towards automatic usability evaluation of spoken dialogue services by user error simulations”. In: *Ninth International Conference on Spoken Language Processing*.

Li, Jiwei et al. (2016). “Deep reinforcement learning for dialogue generation”. In: *arXiv preprint arXiv:1606.01541*.

Yu, Zhou et al. (2016). “Strategy and policy learning for non-task-oriented conversational systems”. In: *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pp. 404–412.

Shah, Pararth et al. (2018). “Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pp. 41–51.

Jaques, Natasha et al. (2019). “Way off-policy batch deep reinforcement learning of implicit human preferences in dialog”. In: *arXiv preprint arXiv:1907.00456*.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press.

Conn, Andrew R, Nicholas IM Gould, and Ph L Toint (2000). *Trust region methods*. Vol. 1. Siam.

Zhu, Chen et al. (2020b). “FreeLB: Enhanced Adversarial Training for Natural Language Understanding”. In:

Nachum, Ofir et al. (2019b). “Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections”. In: *Advances in Neural Information Processing Systems*, pp. 2318–2328.

Liu, Qiang et al. (2018). “Breaking the curse of horizon: Infinite-horizon off-policy estimation”. In: *Advances in Neural Information Processing Systems*, pp. 5356–5366.

Dinan, Emily et al. (2020). “The second conversational intelligence challenge (convai2)”. In: *The NeurIPS’18 Competition*. Springer, pp. 187–208.

Miyato, Takeru et al. (2018). “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.8, pp. 1979–1993.

Shu, Rui et al. (2018). “A dirt-t approach to unsupervised domain adaptation”. In: *arXiv preprint arXiv:1802.08735*.

Zhang, Hongyang et al. (2019a). “Theoretically Principled Trade-off between Robustness and Accuracy”. In: *International Conference on Machine Learning*, pp. 7472–7482.

Hampel, Frank R (1974). “The influence curve and its role in robust estimation”. In: *Journal of the american statistical association* 69.346, pp. 383–393.

Huber, Peter J (2011). *Robust statistics*. Springer.

Raskutti, Garvesh and Sayan Mukherjee (2015). “The information geometry of mirror descent”. In: *IEEE Transactions on Information Theory* 61.3, pp. 1451–1457.

Tarvainen, Antti and Harri Valpola (2017). “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In: *Advances in neural information processing systems*, pp. 1195–1204.

Wang, Alex et al. (2018). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *EMNLP 2018*, p. 353.

Wolf, Thomas et al. (2019a). “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *ArXiv* abs/1910.03771.

Liu, Xiaodong et al. (2020c). “The Microsoft Toolkit of Multi-Task Deep Neural Networks for Natural Language Understanding”. In: *arXiv preprint arXiv:2002.07972*.

Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Liu, Liyuan et al. (Apr. 2020a). “On the Variance of the Adaptive Learning Rate and Beyond”. In: *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*.

Wang, Wei et al. (2019c). “StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding”. In: *arXiv preprint arXiv:1908.04577*.

He, Pengcheng et al. (2019). “A Hybrid Neural Network Model for Commonsense Reasoning”. In: *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pp. 13–21.

Kocijan, Vid et al. (2019). “A Surprisingly Robust Trick for the Winograd Schema Challenge”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4837–4842.

Bowman, Samuel et al. (2015). “A large annotated corpus for learning natural language inference”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642.

Khot, Tushar, Ashish Sabharwal, and Peter Clark (2018). “SciTail: A Textual Entailment Dataset from Science Question Answering”. In: *AAAI*.

Nie, Yixin et al. (2019). “Adversarial NLI: A New Benchmark for Natural Language Understanding”. In: *arXiv preprint arXiv:1910.14599*.

Caruana, Rich (1997). “Multitask learning”. In: *Machine learning* 28.1, pp. 41–75.

Liu, Xiaodong et al. (2015). "Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 912–921.

Liu, Xiaodong, Kevin Duh, and Jianfeng Gao (2018). "Stochastic Answer Networks for Natural Language Inference". In: *arXiv preprint arXiv:1804.07888*.

Zhang, Zhuosheng et al. (2018c). *I Know What You Want: Semantic Learning for Text Comprehension*. arXiv: 1809.02794 [cs.CL].

Radford, Alec et al. (2018). "Language models are unsupervised multitask learners". In:

Williams, Adina, Nikita Nangia, and Samuel Bowman (2018). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1112–1122.

Thorne, James et al. (2018). "FEVER: a large-scale dataset for fact extraction and verification". In: *arXiv preprint arXiv:1803.05355*.

Rockafellar, R Tyrrell (1976). "Monotone operators and the proximal point algorithm". In: *SIAM journal on control and optimization* 14.5, pp. 877–898.

Teboulle, Marc (1997). "Convergence of proximal-like algorithms". In: *SIAM Journal on Optimization* 7.4, pp. 1069–1083.

Eckstein, Jonathan (1993). "Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming". In: *Mathematics of Operations Research* 18.1, pp. 202–226.

Güler, Osman (1991). "On the convergence of the proximal point algorithm for convex minimization". In: *SIAM Journal on Control and Optimization* 29.2, pp. 403–419.

— (1992). "New proximal point algorithms for convex minimization". In: *SIAM Journal on Optimization* 2.4, pp. 649–664.

Parikh, Neal, Stephen Boyd, et al. (2014). "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3, pp. 127–239.

Jiang, Haoming et al. (2020a). "Named Entity Recognition with Small Strongly Labeled and Large Weakly Labeled Data". In: *Preprint*.

Karatay, Deniz and Pinar Karagoz (2015). "User Interest Modeling in Twitter with Named Entity Recognition". In: *Making Sense of Microposts (# Microposts2015)*.

Khalid, Mahboob Alam, Valentin Jijkoun, and Maarten De Rijke (2008). "The impact of named entity normalization on information retrieval for question answering". In: *ECIR*. Springer, pp. 705–710.

Bowden, Kevin et al. (2018). "SlugNERDS: A Named Entity Recognition Tool for Open Domain Dialogue Systems". In: *LREC*.

Yang, Yaosheng et al. (2018). "Distantly supervised ner with partial annotation learning and reinforcement learning". In: *COLING*, pp. 2159–2169.

Shang, Jingbo et al. (Oct. 2018). "Learning Named Entity Tagger using Domain-Specific Dictionary". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2054–2064.

Ni, Jian, Georgiana Dinu, and Radu Florian (2017). "Weakly Supervised Cross-Lingual Named Entity Recognition via Effective Annotation and Representation Projection". In: *ACL*, pp. 1470–1480.

Cao, Yixin et al. (Nov. 2019). "Low-Resource Name Tagging Learned with Weakly Labeled Data". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 261–270.

Lan, Zhenzhong et al. (2020c). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *ICLR*.

Raffel, Colin et al. (2019b). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683. arXiv: 1910.10683.

Li, Qi et al. (2012). "Joint bilingual name tagging for parallel corpora". In: *CIKM*, pp. 1727–1731.

Giannakopoulos, Athanasios et al. (2017). "Unsupervised Aspect Term Extraction with B-LSTM & CRF using Automatically Labelled Datasets". In: *the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 180–188.

Fries, Jason et al. (2017). "Swellshark: A generative model for biomedical named entity recognition without labeled data". In: *arXiv preprint arXiv:1704.06360*.

Sang, Erik F and Fien De Meulder (2003). "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *arXiv preprint cs/0306050*.

Strauss, Benjamin et al. (2016). “Results of the wnut16 named entity recognition shared task”. In: *WNUT*, pp. 138–144.

Jiang, Haoming et al. (2019). “SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization”. In: *arXiv preprint arXiv:1911.03437*.

Zhu, Yukun et al. (2015). “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books”. In: *ICCV*, pp. 19–27.

Vrandečić, Denny and Markus Krötzsch (2014). “Wikidata: a free collaborative knowledgebase”. In: *Communications of the ACM* 57.10, pp. 78–85.

Liu, Liyuan et al. (2020b). “On the Variance of the Adaptive Learning Rate and Beyond”. In: *ICLR*.

Xie, Junyuan, Ross Girshick, and Ali Farhadi (2016a). “Unsupervised deep embedding for clustering analysis”. In: *ICML*, pp. 478–487.

Tjong Kim Sang, Erik F. and Fien De Meulder (2003). “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *Proceedings of CoNLL-2003*. Ed. by Walter Daelemans and Miles Osborne. Edmonton, Canada, pp. 142–147.

Godin, Fréderic et al. (2015). “Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations”. In: *WNUT*, pp. 146–153.

Weischedel, Ralph et al. (2013). “Ontonotes release 5.0 ldc2013t19”. In: *Linguistic Data Consortium, Philadelphia, PA* 23.

Balasuriya, Dominic et al. (2009). “Named entity recognition in wikipedia”. In: *the 2009 Workshop on The People’s Web Meets NLP*, pp. 10–18.

Ratinov, Lev and Dan Roth (2009). “Design challenges and misconceptions in named entity recognition”. In: *CoNLL*, pp. 147–155.

Ma, Xuezhe and Eduard Hovy (2016). “End-to-end sequence labeling via bi-directional lstm-cnns-crf”. In: *ACL*, 1064–1074.

Liu, Angli, Jingfei Du, and Veselin Stoyanov (2019). “Knowledge-augmented language model and its application to unsupervised named-entity recognition”. In: *NAACL*.

Lan, Ouyu et al. (2020a). “Learning to Contextually Aggregate Multi-Source Supervision for Sequence Labeling”. In: *ACL*.

Limsopatham, Nut and Nigel Collier (2016). "Bidirectional LSTM for named entity recognition in Twitter messages". In:

Gururangan, Suchin et al. (2020). "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *arXiv preprint arXiv:2004.10964*.

Lee, Jinhyuk et al. (2020). "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4, pp. 1234–1240.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *ICML*.

Zadrozny, Bianca and Charles Elkan (2001). "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *Icml*. Vol. 1. Citeseer, pp. 609–616.

Carpenter, B (2009). "Coding chunkers as taggers: Io, bio, bmewo, and bmewo+". In: *LingPipe Blog*, p. 14.

Wolf, Thomas et al. (2019b). "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv*, arXiv–1910.

Wei, Chih-Hsuan et al. (2015). "Overview of the BioCreative V chemical disease relation (CDR) task". In: *Proceedings of the fifth BioCreative challenge evaluation workshop*. Vol. 14.

Doğan, Rezarta Islamaj, Robert Leaman, and Zhiyong Lu (2014). "NCBI disease corpus: a resource for disease name recognition and concept normalization". In: *Journal of biomedical informatics* 47, pp. 1–10.

Crichton, Gamal et al. (2017). "A neural network multi-task learning approach to biomedical named entity recognition". In: *BMC bioinformatics* 18.1, p. 368.

Gu, Yu et al. (2020). "Domain-specific language model pretraining for biomedical natural language processing". In: *arXiv preprint arXiv:2007.15779*.

Wang, Yaqing et al. (2020). "Adaptive Self-training for Few-shot Neural Sequence Labeling". In: *arXiv preprint arXiv:2010.03680*.

Du, Jingfei et al. (2020). "Self-training Improves Pre-training for Natural Language Understanding". In: *CoRR* abs/2010.02194. arXiv: 2010.02194.

Mann, Gideon S and Andrew McCallum (2010). "Generalized Expectation Criteria for Semi-Supervised Learning with Weakly Labeled Data." In: *Journal of machine learning research* 11.2.

Ghosh, Aritra, Himanshu Kumar, and PS Sastry (2017). "Robust loss functions under label noise for deep neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1.

Nooralahzadeh, Farhad, Jan Tore Lønning, and Lilja Øvrelid (2019). "Reinforcement-based denoising of distantly supervised NER with partial annotation". In: Association for Computational Linguistics.

Cotterell, Ryan and Kevin Duh (Nov. 2017). "Low-Resource Named Entity Recognition with Cross-lingual, Character-Level Neural Conditional Random Fields". In: *IJCNLP*. Asian Federation of Natural Language Processing, pp. 91–96.

Feng, Xiaocheng et al. (July 2018). "Improving Low Resource Named Entity Recognition using Cross-lingual Knowledge Transfer". In: *IJCAI*, pp. 4071–4077.

Xie, Jiateng et al. (2018). "Neural Cross-Lingual Named Entity Recognition with Minimal Resources". In: *EMNLP*, pp. 369–379.

Rahimi, Afshin, Yuan Li, and Trevor Cohn (2019). "Multilingual NER transfer for low-resource languages". In: *arXiv preprint arXiv:1902.00193*.

Meng, Yu et al. (2018). "Weakly-Supervised Neural Text Classification". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 983–992. ISBN: 9781450360142.

Clark, Kevin et al. (2018). "Semi-Supervised Sequence Modeling with Cross-View Training". In: *EMNLP*.

Yu, Yue et al. (2021). "Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*.

Dodge, Jesse et al. (2020). "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping". In: *CoRR* abs/2002.06305. arXiv: 2002.06305.

Xie, Junyuan, Ross Girshick, and Ali Farhadi (2016b). "Unsupervised Deep Embedding for Clustering Analysis". In: *Proceedings of The 33rd International Conference on Machine Learning*, pp. 478–487.

Meng, Yu et al. (2020). "Text Classification Using Label Names Only: A Language Model Self-Training Approach". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005). “Learning a similarity metric discriminatively, with application to face verification”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1, pp. 539–546.

Taigman, Yaniv et al. (June 2014). “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pereyra, Gabriel et al. (2017). “Regularizing Neural Networks by Penalizing Confident Output Distributions”. In: *CoRR* abs/1701.06548. arXiv: 1701.06548.

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). “Character-level Convolutional Networks for Text Classification”. In: *Advances in Neural Information Processing Systems 28*, pp. 649–657.

Maas, Andrew L. et al. (June 2011). “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150.

Voorhees, Ellen M. and Dawn M. Tice (1999). “The TREC-8 Question Answering Track Evaluation”. In: *Proceedings of The Eighth Text REtrieval Conference*. Ed. by Ellen M. Voorhees and Donna K. Harman. Vol. 500-246.

Liu, Jingjing et al. (2013). “Query understanding enhanced by hierarchical parsing structures”. In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, pp. 72–77.

Krallinger, Martin, Obdulia Rabal, Saber A Akhondi, et al. (2017). “Overview of the BioCreative VI chemical-protein interaction Track”. In: *Proceedings of the sixth BioCreative challenge evaluation workshop*. Vol. 1, pp. 141–146.

Pilehvar, Mohammad Taher and Jose Camacho-Collados (June 2019). “WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1267–1273.

Zhang, Hongyi et al. (2018a). “mixup: Beyond Empirical Risk Minimization”. In: *International Conference on Learning Representations*.

Mukherjee, Subhabrata and Ahmed Hassan Awadallah (2020). “Uncertainty-aware Self-training for Text Classification with Few Labels”. In: *CoRR* abs/2006.15315. arXiv: 2006.15315.

Gal, Yarin and Zoubin Ghahramani (2015). “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference”. In: *CoRR* abs/1506.02158. arXiv: 1506.02158.

Devlin, Jacob et al. (June 2019b). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Levine, Yoav et al. (July 2020). “SenseBERT: Driving Some Sense into BERT”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4656–4667.

Wang, Alex et al. (2019a). “Superglue: A stickier benchmark for general-purpose language understanding systems”. In: *Advances in Neural Information Processing Systems*, pp. 3261–3275.

Maaten, Laurens van der and Geoffrey Hinton (2008). “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Wang, Boxin et al. (2021). “InfoBERT: Improving Robustness of Language Models from An Information Theoretic Perspective”. In: *International Conference on Learning Representations*.

Gunel, Beliz et al. (2021). “Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning”. In: *International Conference on Learning Representations*.

Zhang, Tianyi et al. (2021). “Revisiting Few-sample BERT Fine-tuning”. In: *International Conference on Learning Representations*.

Aghajanyan, Armen et al. (2021). “Better Fine-Tuning by Reducing Representational Collapse”. In: *International Conference on Learning Representations*.

Lison, Pierre et al. (2020). “Named Entity Recognition without Labelled Data: A Weak Supervision Approach”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Mekala, Dheeraj and Jingbo Shang (July 2020). “Contextualized Weak Supervision for Text Classification”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 323–333.

Jiang, Haoming et al. (2021). “Towards Automatic Evaluation of Dialog Systems: A Model-Free Off-Policy Evaluation Approach”. In: *arXiv preprint arXiv:2102.10242*.

Puterman, Martin L (1995). "Markov decision processes: Discrete stochastic dynamic programming". In: *Journal of the Operational Research Society* 46.6, pp. 792–792.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*.

Precup, Doina (2000). "Eligibility traces for off-policy policy evaluation". In: *Computer Science Department Faculty Publication Series*, p. 80.

Voloshin, Cameron et al. (2019). "Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning". In: *arXiv preprint arXiv:1911.06854*.

Horvitz, Daniel G and Donovan J Thompson (1952). "A generalization of sampling without replacement from a finite universe". In: *Journal of the American statistical Association* 47.260, pp. 663–685.

Zhang, Ruiyi et al. (2020b). "Gendice: Generalized offline estimation of stationary values". In: *arXiv preprint arXiv:2002.09072*.

Yang, Mengjiao et al. (2020). "Off-Policy Evaluation via the Regularized Lagrangian". In: *arXiv preprint arXiv:2007.03438*.

Yin, Ming and Yu-Xiang Wang (2020). "Asymptotically efficient off-policy evaluation for tabular reinforcement learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3948–3958.

Duan, Yaqi and Mengdi Wang (2020). "Minimax-Optimal Off-Policy Evaluation with Linear Function Approximation". In: *arXiv preprint arXiv:2002.09516*.

Lagoudakis, Michail G and Ronald Parr (2003). "Least-squares policy iteration". In: *Journal of machine learning research* 4.Dec, pp. 1107–1149.

Mataric, Maja J (1994). "Reward functions for accelerated learning". In: *Machine learning proceedings 1994*. Elsevier, pp. 181–189.

Dai, Bo et al. (2017). "Boosting the actor with dual critic". In: *arXiv preprint arXiv:1712.10282*.

Chen, Zhehui et al. (2018). "On landscape of Lagrangian functions and stochastic search for constrained nonconvex optimization". In: *arXiv preprint arXiv:1806.05151*.

Xie, Tengyang, Yifei Ma, and Yu-Xiang Wang (2019). "Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling". In: *Advances in Neural Information Processing Systems*, pp. 9668–9678.

Uehara, Masatoshi, Jiawei Huang, and Nan Jiang (2019). "Minimax Weight and Q-Function Learning for Off-Policy Evaluation". In: *arXiv*, arXiv–1910.

Jiang, Nan and Lihong Li (2016). "Doubly robust off-policy value evaluation for reinforcement learning". In: *International Conference on Machine Learning*. PMLR, pp. 652–661.

Kallus, Nathan and Masatoshi Uehara (2019). "Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning". In: *arXiv preprint arXiv:1909.05850*.

Papineni, Kishore et al. (2002). "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Zhang, Saizheng et al. (2018b). "Personalizing dialogue agents: I have a dog, do you have pets too?" In: *arXiv preprint arXiv:1801.07243*.

Nachum, Ofir et al. (2019a). "Algaedice: Policy gradient from arbitrary experience". In: *arXiv preprint arXiv:1912.02074*.

Kallus, Nathan and Masatoshi Uehara (2020). "Statistically efficient off-policy policy gradients". In: *International Conference on Machine Learning*. PMLR, pp. 5089–5100.

Rajpurkar, Pranav et al. (Nov. 2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2383–2392.

Warstadt, Alex, Amanpreet Singh, and Samuel R Bowman (2019). "Neural network acceptability judgments". In: *Transactions of the Association for Computational Linguistics* 7, pp. 625–641.

Socher, Richard et al. (2013). "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642.

Cer, Daniel et al. (2017). "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14.

Dolan, William B and Chris Brockett (2005). "Automatically constructing a corpus of sentential paraphrases". In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dagan, Ido, Oren Glickman, and Bernardo Magnini (2006). "The PASCAL Recognising Textual Entailment Challenge". In: *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object*

Classification, and Recognizing Textual Entailment. MLCW'05. Southampton, UK: Springer-Verlag, pp. 177–190. ISBN: 3-540-33427-0, 978-3-540-33427-9.

Bar-Haim, Roy et al. (Jan. 2006). "The second PASCAL recognising textual entailment challenge". In: *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Giampiccolo, Danilo et al. (June 2007). "The Third PASCAL Recognizing Textual Entailment Challenge". In: *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Prague: Association for Computational Linguistics, pp. 1–9.

Bentivogli, Luisa et al. (2009). "The Fifth PASCAL Recognizing Textual Entailment Challenge". In: *In Proc Text Analysis Conference (TAC'09*.

Levesque, Hector, Ernest Davis, and Leora Morgenstern (2012). "The winograd schema challenge". In: *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Loper, Edward and Steven Bird (2002). "NLTK: the natural language toolkit". In: *arXiv preprint cs/0205028*.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation". In: *EMNLP*, pp. 1532–1543.

Wu, Yonghui et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Zhou, Wenxuan et al. (2020). "NERO: A Neural Rule Grounding Framework for Label-Efficient Relation Extraction". In: *Proceedings of The Web Conference 2020*, 2166–2176. ISBN: 9781450370233.

Wu, Shanchan and Yifan He (2019). "Enriching Pre-Trained Language Model with Entity Information for Relation Classification". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2361–2364. ISBN: 9781450369763.

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415*.

Wolf, Thomas et al. (2019c). "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* abs/1910.03771.

Miller, A. H. et al. (2017). “ParlAI: A Dialog Research Software Platform”. In: *arXiv preprint arXiv:1705.06476*.

Finch, Sarah E and Jinho D Choi (2020). “Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols”. In: *arXiv preprint arXiv:2006.06110*.

Deriu, Jan et al. (2020). “Survey on evaluation methods for dialogue systems”. In: *Artificial Intelligence Review*, pp. 1–56.

Brown, Peter F et al. (1992). “An estimate of an upper bound for the entropy of English”. In: *Computational Linguistics* 18.1, pp. 31–40.

Banerjee, Satanjeev and Alon Lavie (2005). “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments”. In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Lin, Chin-Yew (2004). “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*, pp. 74–81.

Galley, Michel et al. (2015). “deltableu: A discriminative metric for generation tasks with intrinsically diverse targets”. In: *arXiv preprint arXiv:1506.06863*.

Rus, Vasile and Mihai Lintean (2012). “An optimal assessment of natural language student input using word-to-word similarity metrics”. In: *International Conference on Intelligent Tutoring Systems*. Springer, pp. 675–676.

Forgues, Gabriel et al. (2014). “Bootstrapping dialog systems with word embeddings”. In: *Nips, modern machine learning and natural language processing workshop*. Vol. 2.

Wieting, John et al. (2015). “Towards universal paraphrastic sentence embeddings”. In: *arXiv preprint arXiv:1511.08198*.

Tao, Chongyang et al. (2017). “Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems”. In: *arXiv preprint arXiv:1701.03079*.

Shimanaka, Hiroki, Tomoyuki Kajiwara, and Mamoru Komachi (2019). “Machine translation evaluation with BERT regressor”. In: *arXiv preprint arXiv:1907.12679*.

Zhang, Tianyi et al. (2019b). “Bertscore: Evaluating text generation with bert”. In: *arXiv preprint arXiv:1904.09675*.

Ghazarian, Sarik et al. (2019). “Better automatic evaluation of open-domain dialogue systems with contextualized embeddings”. In: *arXiv preprint arXiv:1904.10635*.

Li, Junlong et al. (2020). “Task-specific Objectives of Pre-trained Language Models for Dialogue Adaptation”. In: *arXiv preprint arXiv:2009.04984*.

Mehri, Shikib and Maxine Eskenazi (2020). “Unsupervised evaluation of interactive dialog with dialogpt”. In: *arXiv preprint arXiv:2006.12719*.

Gao, Xiang et al. (2020). “Dialogue Response Ranking Training with Large-Scale Human Feedback Data”. In: *arXiv preprint arXiv:2009.06978*.

Lan, Tian et al. (2020b). “PONE: A Novel Automatic Evaluation Metric for Open-Domain Generative Dialogue Systems”. In: *arXiv preprint arXiv:2004.02399*.

Pang, Bo et al. (July 2020). “Towards Holistic and Automatic Evaluation of Open-Domain Dialogue Generation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 3619–3629.

Zhang, Hainan et al. (2020a). “Modeling topical relevance for multi-turn dialogue generation”. In: *arXiv preprint arXiv:2009.12735*.

Yuma, Tsuta, Naoki Yoshinaga, and Masashi Toyoda (July 2020). “uBLEU: Uncertainty-Aware Automatic Evaluation Method for Open-Domain Dialogue Systems”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Online: Association for Computational Linguistics, pp. 199–206.

Zhao, Tianyu, Divesh Lala, and Tatsuya Kawahara (2020). “Designing Precise and Robust Dialogue Response Evaluators”. In: *arXiv preprint arXiv:2004.04908*.

Sai, Ananya B et al. (2020). “Improving Dialog Evaluation with a Multi-reference Adversarial Dataset and Large Scale Pretraining”. In: *Transactions of the Association for Computational Linguistics* 8, pp. 810–827.

Huang, Lishan et al. (2020). “GRADE: Automatic Graph-Enhanced Coherence Metric for Evaluating Open-Domain Dialogue Systems”. In: *arXiv preprint arXiv:2010.03994*.

Sellam, Thibault, Dipanjan Das, and Ankur P Parikh (2020). “BLEURT: Learning Robust Metrics for Text Generation”. In: *arXiv preprint arXiv:2004.04696*.

Hemphill, Charles T, John J Godfrey, and George R Doddington (1990). “The ATIS spoken language systems pilot corpus”. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Williams, Jason et al. (Aug. 2013). “The Dialog State Tracking Challenge”. In: *Proceedings of the SIGDIAL 2013 Conference*. Metz, France: Association for Computational Linguistics, pp. 404–413.

Wang, Jie, Rui Gao, and Hongyuan Zha (2020). “Reliable Off-policy Evaluation for Reinforcement Learning”. In: *arXiv preprint arXiv:2011.04102*.

Wang, Ruosong, Dean P Foster, and Sham M Kakade (2020). “What are the Statistical Limits of Offline RL with Linear Function Approximation?” In: *arXiv preprint arXiv:2010.11895*.

VITA

Haoming Jiang graduated from Shenzhen Middle School in 2013. He received his B.S. Degree in Computer Science and Mathematics from the School of the Gifted Young at University of Science and Technology of China (USTC) in 2017. Afterwards, his interest was drawn towards machine learning, and he joined the Ph.D. program in the School of Industrial and Systems Engineering at Georgia Institute of Technology.